NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA)
SCHOOL OF CIVIL ENGINEERING
INSTITUTE OF STRUCTURAL ANALYSIS AND SEISMIC RESEARCH

INNOVATIVE COMPUTATIONAL TECHNIQUES FOR THE OPTIMUM STRUCTURAL DESIGN CONSIDERING UNCERTAINTIES

PhD Dissertation

by

Vagelis Plevris

Advisor:

Professor Manolis Papadrakakis

June 2009

Final version

National Technical University of Athens
School of Civil Engineering
Institute of Structural Analysis and Seismic Research

Innovative Computational Techniques for the Optimum Structural Design Considering Uncertainties

A dissertation submitted to the School of Civil Engineering of National Technical University of Athens in partial fulfillment of the requirements for the degree of Doctor of Philosophy

by Vagelis Plevris

Advisor:
Professor Manolis Papadrakakis

Athens,
June 2009

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Πολιτικών Μηχανικών
Εργαστήριο Στατικής και Αντισεισμικών Ερευνών

Προηγμένες Υπολογιστικές Τεχνικές Βέλτιστου Σχεδιασμού Κατασκευών με Αβεβαιότητες

Η διατριβή αυτή υποβλήθηκε στη Σχολή Πολιτικών Μηχανικών του Εθνικού Μετσοβίου Πολυτεχνείου προς μερική εκπλήρωση των απαιτήσεων για την απόκτηση Διδακτορικού τίτλου σπουδών

από τον Βαγγέλη Πλεύρη

Επιβλέπων:
Καθηγητής Μανόλης Παπαδρακάκης

Αθήνα,
Ιούνιος 2009

***Dedicated to the memory of my parents**:*

***My mother**, Popi Plevri,*

who passed away on June 24, 2009,

to honor her love and support throughout my life.

***My father**, Manolis Plevris*

who passed away on January 12, 2005,

for his love and his positive influence on my life.

This is the final version of the PhD dissertation,
after the examination of July 10, 2009.

PhD Examination Committee

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Manolis Papadrakakis
Professor
(Principal Advisor)
School of Civil Engineering
National Technical University of Athens

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Christoforos Provatidis
Associate Professor
(Member of advisory committee)
School of Mechanical Engineering
National Technical University of Athens

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yiannis Tsompanakis
Assistant Professor
(Member of advisory committee)
Department of Applied Sciences
Technical University of Crete

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Konstantinos Spyrakos
Professor
School of Civil Engineering
National Technical University of Athens

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Andreas–Georgios Stafylopatis
Professor
School of Electrical and Computer Engineering
National Technical University of Athens

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Matthew Karlaftis
Assistant Professor
School of Civil Engineering
National Technical University of Athens

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Nikos D. Lagaros
Lecturer
School of Civil Engineering
National Technical University of Athens

Abstract

Uncertainties in structural mechanics, and in particular in the phase of analysis and design, can play an extremely important role, affecting not only the safety and reliability of structures and their mechanical components, but also the quality of their performance. The response of a structural system may be very sensitive to uncertainties in the material properties, manufacturing conditions, external loading and analytical or numerical modeling. In order to account for these issues, stochastic analysis methods have been developed over the last decades. The optimum result obtained by a deterministic optimization formulation that ignores scatter of any kind of the parameters affecting its response has limited value and reliability, as it can be severely affected by the uncertainties that are inherent in the model. The deterministic optimum can be associated with unaccepted probabilities of failure, or it can be vulnerable to slight variations of some uncertain parameters. The development of probabilistic analysis methods over the last two decades has stimulated the interest for considering also randomness and uncertainty in the formulation of structural design optimization problems. In order to account for uncertainties in a structural optimization framework, probabilistic-based formulations of the optimization problem have to be used, utilizing stochastic simulation and probabilistic analysis.

The goal of the thesis is to unify the concepts of probability-based safety analysis and structural optimization and provide the necessary numerical tools to deal with optimization problems considering uncertainties. This goal is addressed by developing algorithms for solving the probabilistic structural optimization problems encountered. In order to deal with these problems efficiently, various algorithms and methodologies have to be used, such as efficient single- and multi-objective optimizers and efficient stochastic problems formulations for the stochastic analysis process. Despite the advances on these issues, the computational cost for considering the uncertainties in a structural design optimization problem remains extremely large, especially for real-world large-scale problems with many design and/or random variables. To alleviate the computational burden, the implementation of *Neural Network* (NN) metamodels is also proposed in this thesis for further reducing the computational cost, providing acceptable numerical results at an affordable computational time.

The dissertation consists of nine chapters in total, plus the bibliography and three appendices. It is organized as follows: following the introduction of Chapter 1, Chapter 2 deals with the concept of uncertainty in structural engineering in general. Chapter 3 presents the formulation of single objective optimization problems, while Chapter 4 discusses the multi-objective optimization problem. The basics of Neural Networks and their implementation in structural engineering are presented in Chapter 5. Chapter 6 discusses the problem of structural optimization considering uncertainties, where the basic problems of this kind, namely the *Reliability-Based Design Optimization* (RBDO),

the *Robust Design Optimization* (RDO) and the combination *Reliability-based Robust Design Optimization* (RRDO) problems are presented, among others.

The numerical applications of the dissertation are divided into two parts, A and B, presented in Chapters 7 and 8, respectively. Part A (Chapter 7) contains the deterministic optimization test examples, where uncertainties are not taken into account. In Part B (Chapter 8), the probabilistic optimization test examples are discussed, where uncertainties play a significant role.

Chapter 9 contains the conclusions, the original contribution of the thesis, and directions for future research. Finally, the bibliography is presented followed by three appendices: Appendix A, containing the notation and symbols used in the dissertation; Appendix B with the acronyms and abbreviations used; and Appendix C with a listing of publications by the author.

Περίληψη

Οι αβεβαιότητες στη δομοστατική μηχανική, και ιδιαίτερα κατά τη φάση της ανάλυσης και του σχεδιασμού μιας κατασκευής, μπορούν να παίξουν σημαντικό ρόλο, επηρεάζοντας όχι μόνο την ασφάλεια και την αξιοπιστία της κατασκευής και των μερών από τα οποία αποτελείται, αλλά και την ποιότητα των επιδόσεών της. Η απόκριση ενός δομικού συστήματος μπορεί να είναι ιδιαίτερα ευαίσθητη στις αβεβαιότητες των ιδιοτήτων των υλικών, των συνθηκών της κατασκευής, των εξωτερικών φορτίων και των αναλυτικών ή αριθμητικών μεθόδων που χρησιμοποιήθηκαν για την προσομοίωση του φυσικού προβλήματος. Για να ληφθούν υπόψη αυτές οι αβεβαιότητες, έχουν αναπτυχθεί τις τελευταίες δεκαετίες κατάλληλες μέθοδοι στοχαστικής ανάλυσης των κατασκευών. Το βέλτιστο αποτέλεσμα που προκύπτει από μία προσδιοριστική (αιτιοκρατική) θεώρηση της διαδικασίας βελτιστοποίησης η οποία αγνοεί τη διασπορά των τιμών των παραμέτρων που επηρεάζουν την απόκριση της κατασκευής, έχει περιορισμένη αξία και αξιοπιστία, καθώς μπορεί να επηρεαστεί σημαντικά από εγγενείς αβεβαιότητες τόσο του φυσικού προβλήματος όσο και του αριθμητικού προσομοιώματος. Το προσδιοριστικό βέλτιστο μπορεί επομένως να σχετίζεται με μη αποδεκτή τιμή της πιθανότητας αστοχίας, ή μπορεί να είναι ιδιαίτερα ευαίσθητο σε σχετικά μικρές διακυμάνσεις κάποιων παραμέτρων. Η ανάπτυξη στοχαστικών - πιθανοτικών μεθόδων ανάλυσης κατά τις δύο τελευταίες δεκαετίες έχει κεντρίσει το ενδιαφέρον των ερευνητών για την εισαγωγή των εννοιών της αβεβαιότητας και της τυχηματικότητας στις διατυπώσεις των προβλημάτων βέλτιστου σχεδιασμού των κατασκευών. Για να ληφθούν υπόψη οι αβεβαιότητες στα πλαίσια ενός προβλήματος βελτιστοποίησης, πρέπει να χρησιμοποιηθούν διατυπώσεις βασισμένες στην πιθανοτική φύση του προβλήματος, χρησιμοποιώντας τη στοχαστική ανάλυση και τη θεωρία πιθανοτήτων.

Ο στόχος της διατριβής είναι η ενοποιημένη αντιμετώπιση της στοχαστικής ανάλυσης και του βέλτιστου σχεδιασμού των κατασκευών και η παροχή των απαραίτητων υπολογιστικών εργαλείων για την επίλυση του προβλήματος της βελτιστοποίησης των κατασκευών με θεώρηση αβεβαιοτήτων. Ο στόχος αυτός επιτυγχάνεται με την ανάπτυξη κατάλληλων αλγορίθμων για την επίλυση του στοχαστικού προβλήματος βελτιστοποίησης. Για να αντιμετωπιστούν αυτά τα προβλήματα με αποδοτικό τρόπο, πρέπει να χρησιμοποιηθούν διάφοροι αλγόριθμοι και μεθοδολογίες βέλτιστου σχεδιασμού, τόσο για προβλήματα μίας όσο και για προβλήματα πολλών αντικειμενικών συναρτήσεων, καθώς και επαρκείς μεθοδολογίες για την αντιμετώπιση του στοχαστικού προβλήματος. Παρά την εφαρμογή προχωρημένων μεθοδολογιών για την αντιμετώπιση των παραπάνω προβλημάτων, το υπολογιστικό κόστος για τη θεώρηση των αβεβαιοτήτων σε ένα πρόβλημα βελτιστοποίησης παραμένει εξαιρετικά υψηλό, ειδικά για προβλήματα πραγματικών κατασκευών μεγάλης κλίμακας με πολλές μεταβλητές σχεδιασμού ή/και αβέβαιες παραμέτρους. Για τον περιορισμό του προβλήματος αυτού και τη μείωση του υπολογιστικού κόστους, στην παρούσα διατριβή προτείνεται η εφαρμογή Νευρωνικών Δικτύων (*Neural Networks*, NNs)

ως μετα-μοντέλων (metamodels), τα οποία δίνουν ικανοποιητικές λύσεις με ιδιαίτερα χαμηλό υπολογιστικό κόστος.

Η διατριβή αποτελείται συνολικά από εννέα κεφάλαια, τη βιβλιογραφία και τρία παραρτήματα. Η διάρθρωσή της έχει ως εξής: Μετά από την εισαγωγή του 1ου Κεφαλαίου, το 2ο Κεφάλαιο εξετάζει το θέμα των αβεβαιοτήτων σε προβλήματα δομοστατικής μηχανικής. Το 3ο Κεφάλαιο παρουσιάζει τη διατύπωση του προβλήματος βελτιστοποίησης με μία αντικειμενική συνάρτηση (single-objective optimization), ενώ το 4ο Κεφάλαιο εξετάζει το πρόβλημα της βελτιστοποίησης με πολλές αντικειμενικές συναρτήσεις (multi-objective optimization). Τα βασικά στοιχεία των Νευρωνικών Δικτύων (Neural Networks) και οι εφαρμογές τους σε προβλήματα δομοστατικής μηχανικής παρουσιάζονται στο 5ο Κεφάλαιο. Το 6ο Κεφάλαιο εξετάζει το πρόβλημα της βελτιστοποίησης κατασκευών με θεώρηση αβεβαιοτήτων, στο οποίο παρουσιάζονται μεταξύ άλλων οι κυριότερες διατυπώσεις αυτών των προβλημάτων, το πρόβλημα του Βέλτιστου Σχεδιασμού με βάση την Αξιοπιστία (*Reliability-Based Design Optimization*, RBDO), το πρόβλημα του Εύρωστου Βέλτιστου Σχεδιασμού (*Robust Design Optimization*, RDO) και το συνδυασμένο πρόβλημα του Εύρωστου Σχεδιασμού με βάση την Αξιοπιστία (*Reliability-based Robust Design Optimization*, RRDO).

Οι αριθμητικές εφαρμογές της διατριβής είναι χωρισμένες σε δύο ενότητες - Μέρη Α και Β, τα οποία παρουσιάζονται στο 7ο και 8ο Κεφάλαιο, αντίστοιχα. Το Μέρος Α (7ο Κεφάλαιο) περιέχει τις προσδιοριστικές αριθμητικές εφαρμογές, όπου οι αβεβαιότητες δεν λαμβάνονται υπόψη στο αριθμητικό προσομοίωμα. Στο Μέρος Β (8ο Κεφάλαιο) εξετάζονται οι πιθανοτικές αριθμητικές εφαρμογές, όπου οι αβεβαιότητες διαδραματίζουν δεσπόζοντα ρόλο.

Το 9ο Κεφάλαιο περιέχει τα συμπεράσματα της διατριβής, την πρωτότυπη συνεισφορά της και κατευθύνσεις για μελλοντική έρευνα. Τέλος, παρουσιάζεται η βιβλιογραφία και τρία παραρτήματα: Το Παράρτημα Α περιέχει τη σημειογραφία και τους μαθηματικούς συμβολισμούς που υιοθετήθηκαν, το Παράρτημα Β περιέχει τα ακρωνύμια και τις συντμήσεις που χρησιμοποιήθηκαν, και το Παράρτημα C περιέχει μια αναλυτική λίστα με τις δημοσιεύσεις του συγγραφέα.

Acknowledgements

I am glad to have the opportunity to thank a number of people who, in many different ways, contributed to the completion of this dissertation.

First and foremost, I am deeply grateful to my advisor, Professor **Manolis Papadrakakis**, for his excellent guidance, his invaluable support and his confidence in me throughout all these years of my PhD research. His love and unstinting devotion to science, his great scientific inspiration, his continuous availability in spite of the busy schedule and his enthusiastic encouragement towards students are qualities of great significance for an advisor and are highly appreciated. Working with him has been a distinct privilege for me, which I hope to maintain also as a member of his research team during the postdoctoral research period.

I would also like to express my deepest thanks to the other two members of the PhD advisory committee: Associate Professor **Yiannis Tsompanakis** from the Department of Applied Sciences of the Technical University of Crete, for his great support, encouragement and friendship and Associate Professor **Christoforos Provatidis** of the School of Mechanical Engineering, National Technical University of Athens (NTUA) for his kindness, support and availability whenever necessary. I also thank them for their time for the careful reading of the manuscript and their valuable comments and suggestions which contributed to improving the quality of the dissertation.

Last from the academic community, but certainly not least, I would like to thank Lecturer **Nikos Lagaros** of the School of Civil Engineering, NTUA for his continuous support and true friendship since my very early research steps. Our fruitful discussions have always been a great source of inspiration for me. His feedback and valuable suggestions on technical issues were of vital importance in the endeavor of achieving the goals of the dissertation. His research work and academic accomplishments give a continuous motivation to us, younger researchers.

Finally, I am grateful to my beloved wife **Niki** for her understanding and her wholeheartedly support of my research activity with patience, without any complaint. Her love and encouragement is a powerful source of inspiration and energy for me in the pursuit of my research goals, while her companionship always helps me lead my thoughts to other important aspects of life, as well.

Athens, June 2009

Vagelis Plevris

This research work has been funded by the project PENED 2003. The project is part of the Operational Program “Competitiveness” (measure 8.3) of the 3rd Community Support Program and is co-funded, 75% of public expenditure through EC - European Social Fund, 25% of public expenditure through the Greek Ministry of Development - General Secretariat of Research and Technology and through private sector.

Η παρούσα εργασία πραγματοποιήθηκε και χρηματοδοτήθηκε στα πλαίσια του προγράμματος ΠΕΝΕΔ 2003, το οποίο συγχρηματοδοτείται κατά 75% από την Ευρωπαϊκή Ένωση – Ευρωπαϊκό Κοινωνικό Ταμείο, κατά 25% από το Ελληνικό Δημόσιο – Υπουργείο Ανάπτυξης – Γενική Γραμματεία Έρευνας και Τεχνολογίας και από τον Ιδιωτικό Τομέα, στο πλαίσιο του Μέτρου 8.3 του Επιχειρησιακού Προγράμματος «Ανταγωνιστικότητα» – Γ΄ Κοινοτικό Πλαίσιο Στήριξης.

Table of Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

1.1 Motivation

The primal engineering task during the design of any structural system is to minimize its construction and operational costs and improve its structural performance. Improvements during the design stage can be achieved either by using simple design rules based on experience, or by an automated way using structural optimization procedures. Taking into account the complexity of a structural optimization problem, it is obvious that finding the mathematical global optimum solution may not be an easy task.

In engineering problems, uncertainty is inherent and the scatter of parameters from their nominal values is unavoidable. Uncertainties in structural mechanics, and in particular in the phase of analysis and design, can play an extremely important role, affecting not only the safety and reliability of structures and their mechanical components, but also the quality of their performance. Under given circumstances, the response of a structural system can be very sensitive to uncertainties in the material properties, manufacturing conditions, external loading and analytical or numerical modeling. Stochastic analysis methods have been developed significantly over the last decades in order to account for the uncertainty encountered in structural mechanics.

The optimum result obtained by a deterministic optimization formulation that ignores scatter of any kind of the parameters affecting its response has limited value, as it can be severely affected by the uncertainties that are inherent in the model. The deterministic optimum can be associated with unacceptable probabilities of failure, or it can be quite vulnerable to slight variations of some uncertain parameters. Consequently, a deterministically optimum design may result in an infeasible design. In real-world conditions the significance of any “optimum” solution would be limited if the uncertainties involved in the geometric and material description of the structure as well as in the loading conditions are not taken into consideration. This is because real-world structures have always imperfections which induce deviations from the nominal state assumed at the analysis phase by the design codes. The development by the scientific community of probabilistic analysis methods over the last two decades has stimulated the interest for considering

also randomness and uncertainty in the formulation of the structural design optimization problem (Schuëller 2005; Tsompanakis et al. 2008). In order to account for uncertainties in a structural design optimization framework, probabilistic-based formulations of the optimization problem have to be used, utilizing stochastic simulation and probabilistic analysis.

The probabilistic-based design optimization methodologies can be widely classified in the following two generic formulations:

i. *Robust Design Optimization* (RDO);

ii. *Reliability-Based Design Optimization* (RBDO).

RDO methods primarily seek to minimize the influence of random variations of the nominal structural dimensions, material parameters and loading on the response of the structure.

On the other hand, the main goal of RBDO methods is to find the optimum design, which at the same time satisfies the objective of the minimum weight in conjunction with limitations on the allowable probability of failure or the probability of exceeding certain characteristic structural response quantities or the problem's constraints.

1.2 Objectives and scope

The goal of the thesis is to explore the available methodologies on the subject, unify the concepts of probability-based safety analysis and structural optimization and provide innovative numerical tools to deal with optimization problems considering uncertainties. This goal is addressed by developing algorithms and methodologies for efficiently solving the RBDO and RDO problems, as well as the combined *Reliability-based Robust Design Optimization* (RRDO) problem.

In order to address these problems efficiently, various algorithms and methodologies have to be used. First, a single-objective optimization algorithm is required, capable of finding the global optimum, without being trapped in local optima, with a satisfactory convergence rate and consequently not requiring excessive computational effort. For the stochastic analysis part of the methodology, special care is required in order to calculate the statistical quantities that are affected by the random variables of the model. For the multi-objective optimization problem encountered in the RDO formulation or in RBDO formulations considering multiple objectives, efficient multi-objective optimization algorithms have to be implemented, able to provide a complete and detailed Pareto Front.

The computational cost for considering the uncertainties in a structural design optimization problem can be enormous, especially when real-world large-scale structures with many design variables and/or random variables are considered. To alleviate the computational burden, it is necessary to use efficient optimization algorithms and efficient sampling techniques for the stochastic analysis process. In many practical cases, even these techniques prove not to be enough. For this reason, in this thesis *Neural Network*

(NN) metamodels are implemented for further reducing the computational cost, providing acceptable numerical results at very low computational cost.

All the issues described above, in the two previous paragraphs, have been addressed in the thesis, as will be described in detail in the following chapters. Furthermore, via numerical applications of real-world large scale structures the proposed computational framework is evaluated and tested. The original contribution of the thesis is presented in detail in Section 9.1 of the Conclusions (Chapter 9).

1.3 Organization and outline

The thesis consists of nine chapters in total, plus the bibliography and three appendices at the end of it. Its structure is organized as follows:

Chapter 1 is the introduction of the dissertation which provides a general description of the motivation, the goals pursued, as well as a brief description of the contents of each chapter.

Chapter 2 deals with the concept of uncertainty in structural engineering in general. The notions of reliability, failure, the performance function of a structural system and structural resistance and demand are discussed. Various methodologies for addressing the stochastic analysis problem are also discussed, namely the *First- and Second-Order Reliability Methods* (FORM/SORM), the *Response Surface Method* (RSM) and the *Monte Carlo Simulation* (MCS) method, highlighting the strengths and drawbacks of every methodology. Sampling methods for MCS are also discussed, such as the *Latin Hypercube Sampling* (LHS), *Importance Sampling* (IS), and other methodologies.

Chapter 3 discusses the notion of single objective optimization. First, the concept of optimum structural design is presented, followed by the formulation of a single objective optimization problem and some necessary definitions. Various methods for solving the problem are presented, including mathematical programming methods and in particular the SQP method, evolutionary methods and in particular the *Evolution Strategies* (ES) method for both continuous and discrete problems. The idea of cascade optimization is illustrated, as well as the method of *Particle Swarm Optimization* (PSO) and the proposed hybrid PSO-SQP methodology which combines the local search of the SQP mathematical optimizer with the global search of PSO.

Chapter 4 discusses the issue of multi-objective optimization. First, the concept of multi-objective optimization is presented, followed by the formulation of the multi-objective optimization problem and some necessary definitions. The concepts of local and global Pareto optimality, domination and non-domination, conflict and criteria, search and decision making are also discussed. Various methods for solving the multi-objective optimization problem are presented. The standard methods include the *Linear Weighting Method* (LWM), *Distance Function Method* (DFM) and *Constraint Method* (CM). Two ES-based multi-objective optimization methodologies are proposed, namely the ESMO algorithm and the CEATm cascade evolutionary algorithm.

Neural Networks and their applications in structural engineering are presented in Chapter 5. First a historical background is given, followed by the description of biological neural networks, and their comparison with artificial NNs. The characteristics of NNs and their use as metamodels are discussed. Neural network elements, transfer functions, network topologies and NN training are also presented. The back-propagation training algorithm is described in detail and finally some problems that may arise with NNs are discussed.

In Chapter 6 the problem of structural optimization considering uncertainties is discussed, where the two basic problems, namely the RBDO and RDO problems are presented among others. The concepts of reliability and robustness, as well as the formulations of the RBDO and RDO problems are discussed, followed by a presentation of the combined RRDO formulation. Finally, the formulation and application of the proposed NN methodologies for the solution of the RBDO, RDO and RRDO problems are presented.

The numerical applications of the dissertation are divided into two parts, A and B, presented in Chapters 7 and 8, respectively. Part A of the numerical applications (Chapter 7) discusses the deterministic optimization cases, where the uncertainties are not taken into account. The chapter is divided into two sections, with five test examples in total: In the first section (Section 7.1), two multi-objective optimization test examples are examined, using either standard methods or the proposed ESMO algorithm for solving the multi-objective optimization problem. In the second section (Section 7.2), three single-objective *Particle Swarm Optimization* (PSO) examples are considered, namely a plane truss and two space trusses, using either the proposed enhanced PSO methodology for constrained structural optimization or the proposed hybrid PSO-SQP methodology.

Part B of the numerical applications (Chapter 8) discusses the probabilistic optimization cases, where uncertainties play a significant role. In this chapter, ten test examples are examined in total. The chapter is divided into five sections: In the first section (Section 8.1), two Robust Design Optimization test examples are considered, implementing standard methods for solving the multi-objective optimization problem. In the second section (Section 8.2), two RDO test examples are considered, using the proposed non-dominant CEATm methodology for solving the multi-objective optimization problem. In the third section (Section 8.3), two Reliability-Based Design Optimization test examples are considered, using NN predictions to reduce the computational cost. In the fourth section (Section 8.4), two Reliability-based Robust Design Optimization test examples are considered. In the fifth section (Section 8.5), two RRDO test examples are considered, using NN predictions in order to reduce the computational cost.

In Chapter 9 the conclusions of the research work are presented. The original contribution of the thesis is clearly stated in Section 9.1. The overall conclusions are presented in Section 9.2. Natural extensions of this work and ideas for future work on the subject of the thesis are given in Section 9.3 of the dissertation.

Finally, the bibliography is presented in a parenthetical author-date referencing system, followed by three appendices: Appendix A, containing the notation and symbols used in the dissertation; Appendix B with the acronyms and abbreviations used; and Appendix C with a listing of publications by the author.

Chapter 2

2 Uncertainty in Structural Engineering

2.1 Theoretical approaches to uncertainty

In the past, natural science, which arose from the mathematical interpretation of natural phenomena, showed a trend to interpret the random results of experiments as a deficiency of the mathematical models rather than as a property of nature itself. In those times, uncertainty was rejected as a natural phenomenon because of the enthusiastic illusion of a science being able to provide exact answers. The foremost example of this deterministic world-view was Newtonian physics and classical mechanics as developed by Galileo and Newton.

However, in later times, the introduction of mathematical models for probability and randomness became an absolute necessity in order to explain physical phenomena in thermodynamics and quantum mechanics. From that point on, the old paradigm of an exact science was abandoned in those areas where the evidence and the magnitude of randomness could no longer be ignored.

Two broad types of uncertainties can be considered in general: (i) *aleatory* uncertainty; and (ii) *epistemic* uncertainty. The word aleatory derives from the Latin word *alea*, which means the rolling of dice. Thus, an aleatory uncertainty is one that is presumed to be the intrinsic randomness of a phenomenon arising because of natural, unpredictable variation in the performance of the system under study. The word epistemic derives from the Greek word «*επιστήμη*», which means science. Thus, an epistemic uncertainty is one that is presumed as being caused by lack of knowledge (or data) about the behavior of the system. Most problems of engineering interest involve both types of uncertainties. The distinction between these two types can be useful in engineering analysis because epistemic uncertainty is reducible. Although some have suggested that a clear distinction between the two types can be made, in the modeling phase it is often difficult to determine whether a particular uncertainty should be put in the aleatory category or the epistemic one and thus the distinction is rather determined by our modeling choices (Der Kiureghian and Ditlevsen 2009). It has been found that both aleatory and epistemic un-

certainty can be treated and analyzed, either separately or combined, using probability theory and statistics.

2.2 Uncertainty in structural engineering

Uncertainties in structural mechanics, analysis and design play an extremely important role. They affect not only the safety and reliability of structures and mechanical components, but also the quality of their performance. Structural engineering requires safety levels that correspond to extremely low probabilities of significant consequences on the structures. Although this has been mankind's prime structural safety requirement for centuries, the means to achieve it has varied widely over time. In an effort to increase safety and structural reliability, safety factors were adopted by code committees in the 1970s in a subjective manner - without a probability basis - and they applied reasonably well to standard common structures. The factors had developed through experience and had been adjusted over the years as confidence developed in the various building methods and systems. When confidence in a system was high and good performance had been shown over the years, the safety factors were gradually reduced by small increments over a number of versions of the applicable code. On the other hand, when accidents or failures occurred, there was a corresponding increase in safety factors. The codes we use today for structural engineering design needs have been largely formed based on this slow, adaptive process.

The trial and error process described above, for the determination of safety factors, is slow and costly and it is quite incapable to adapting to new technologies and new environments in time. As we enter into periods of rapid technology developments, this adaptive method has become unable to account for our increasing needs. Probability-based methods, with the means to apply measures to uncertainty, are the obvious choice for the development of safety factors for these new technologies, providing the means to accommodate new loadings, materials and systems and to drive the appropriate information acquisition to the proper design of such systems.

Nowadays, although there are fields of science where the consideration of randomness is well established, such as quantum mechanics and other branches of modern physics, structural engineering practice follows the trend of classical mechanics not to include uncertainty models in the design process. Probability theory is the logical basis for dealing with uncertainty, thus it should be the basis to structural safety. Despite the fact that over the past 50 years there have been many contributions to the development of the field of structural safety using probability theory, statistics, decision analysis, fuzzy logic and others, widespread acceptance of these concepts by the design community has not occurred until recently (Sexsmith 1999). It is a paradox that structural engineers, on the one hand, do not include probabilities into their calculations, but on the other hand, have long before recognized the importance of uncertainties in the design practice, by using safety factors of several kinds and statistical analysis of experiments for calibrating various code specified parameters (Hurtado 2008).

It can be said that randomness has been in fact considered in structural design in the past, but not in a systematic manner from an analytical - mathematical point of view. While in conventional, deterministic procedures the qualitative assessment of uncertainties is considered to be sufficient, more modern developments concentrate on their rational assessment, i.e. by quantification. This is accomplished by applying methods of statistics and probability and more recently also methods based on fuzzy sets. The fields which emerged from those developments are denoted as *Computational Stochastic Mechanics* as well as *Structural Reliability*.

It should be noted that the basic objective of these methods is not only to account for the probabilities, but mainly to make decisions about structural safety issues, thus probabilities are to be used in a decision making context. It is obvious that the reliability requires a scientifically-oriented calculation, whereas safety factors are a mere practical tool for producing a qualified product. Probability-based safety analysis should become the basis for safety factors in codes of practice and standards, and it is increasingly used to set structural safety requirements for specific structural systems. Its application is rational, in the sense that it uses probability theory to deal with uncertainty. It permits the code committees and individuals responsible for setting safety standards, with the means to be accountable. It permits the evolution of safety standards to proceed by adapting to new information without waiting for unfortunate events to occur in order to trigger changes in safety levels, as was the case in the past. Therefore, in the near future, probability-based safety analysis is bound to move into the mainstream of structural engineering practice.

2.3 Reliability analysis of structures

2.3.1 Definition of failure

Although its definition may seem obvious, the term *failure* means different things to different people. One can claim that a structure fails if it cannot perform its intended function. However, this is a vague definition because the desirable function of the structure has not been specified exactly. In structural reliability analysis, the concept of a *limit state* is used in order to define failure. A limit state is a boundary between desired and undesired performance of a structure. This boundary is often represented mathematically by a *limit state function* or *performance function*. Three broad types of limit state functions can be considered in general: (i) *Ultimate Limit States* (ULSs), mostly related to the loss of load-carrying capacity; (ii) *Serviceability Limit States* (SLSs), related to gradual deterioration, loss of user's comfort under routine conditions, or maintenance costs; and (iii) *Fatigue Limit States* (FLSs), related to the accumulation of damage and eventual loss of strength under repeated loads.

2.3.2 The notion of the performance function

The design of a structure requires the verification of a certain number of rules resulting from the knowledge of mechanics and the experience of the designer and the constructor. These rules come from the necessity to limit loading effects such as stresses and displacements. Each rule represents an elementary event and the occurrence of several events leads to a failure scenario for the structure. In addition to deterministic variables used in the model, the uncertainties are modeled by *stochastic variables* (or *random variables*) affecting the failure scenario. Each stochastic variable is described by statistical information on its value, typically by a given *Probability Density Function* (PDF) or by the type of PDF and some statistical parameters (generally the mean value and the standard deviation).

In the present thesis, the random variables are in general denoted by an underlined lower-case letter. Let $\underline{\boldsymbol{x}} = [\underline{x}_1,...,\underline{x}_m]^{\mathrm{T}}$ be a real-valued vector of *m* random variables (random parameters) of the structural model. A realization of the vector of the random variables would be denoted as vector $\boldsymbol{x} = [x_1,...,x_m]^{\mathrm{T}}$ without underlining. The safety is defined as the state where the structure is able to fulfill all the operating requirements, mechanical and serviceability, for which it is designed, during its lifetime. To evaluate the failure probability with respect to a given failure scenario, the *performance function* $\underline{g} = g(\underline{\boldsymbol{x}})$ (known also as the *limit state function* or the *safety margin*) is defined by the condition of good operation of the structure. The limit between the state of failure $g(\underline{\boldsymbol{x}}) \le 0$ and the state of safety $g(\underline{\boldsymbol{x}}) > 0$ is known as the *limit state surface* $g(\underline{\boldsymbol{x}}) = 0$. The "safety" domain in $\mathbb{R}^m$ can be defined as:

$$\mathcal{D}_{\mathrm{s}} = \{\boldsymbol{x} \in \mathbb{R}^m \mid g(\boldsymbol{x}) > 0\} \tag{2.1}$$

And the "failure" domain in $\mathbb{R}^m$ can be defined as:

$$\mathcal{D}_{\mathrm{f}} = \{\boldsymbol{x} \in \mathbb{R}^m \mid g(\boldsymbol{x}) \le 0\} \tag{2.2}$$

Given the performance function $\underline{g} = g(\underline{\boldsymbol{x}})$, it is possible to evaluate the probability of failure by integrating the joint PDF of all the random variables over the failure domain

$$p_{\mathrm{f}}(\underline{x}) = \int_{\mathcal{D}_{\mathrm{f}}} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \tag{2.3}$$

where $f(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}$ is the joint PDF of all the random variables that satisfies the conditions

$$f(\boldsymbol{x}) > 0, \quad \int_{\mathbb{R}^m} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1 \tag{2.4}$$

In many cases, the performance function $\underline{g} = g(\underline{\boldsymbol{x}})$ can be written as the margin between two other random variables, namely the *structural resistance* $\underline{r} = r(\underline{\boldsymbol{x}})$ (or *structural capacity*) and the *load effect* $\underline{s} = s(\underline{\boldsymbol{x}})$ (or *structural demand*) as follows:

$$\underline{g} = \underline{r} - \underline{s} \tag{2.5}$$

2.3.3 Structural resistance and demand as independent normal variables

Consider the special case where $\underline{r}$ and $\underline{s}$ are two independent random variables that both follow normal distributions with mean values μ_r and μ_s, standard deviations σ_r and σ_s and PDFs $f_r(x)$ and $f_s(x)$, respectively. The PDF $\varphi(x)$ for the *standard normal distribution* with a zero mean (μ=0) and a variance (standard deviation squared) of one (σ^2=1) is given by the formula

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{2.6}$$

while the *Cumulative Distribution Function* (CDF) $\Phi(x)$ for the standard normal distribution of one variable is given by

$$\Phi(x) = \int_{-\infty}^{x} \varphi(x)\mathrm{d}x = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{x^2}{2}\right)\mathrm{d}x \tag{2.7}$$

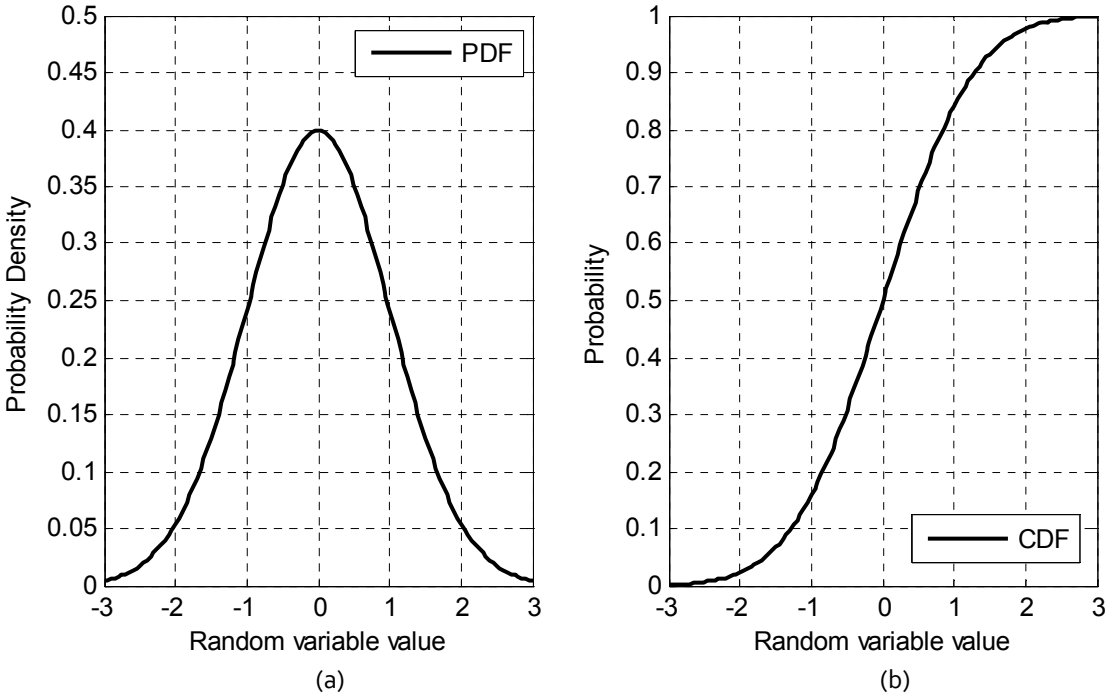


Figure 2.1 Standard normal distribution: (a) PDF and (b) CDF.

Figure 2.1 shows the PDF and the CDF for the standard normal distribution of one variable, plotted for a distance 3 times the standard deviation from the mean value, which accounts for about 99.7% of the set of random values.

The PDF $\varphi_{\mu,\sigma^2}(x)$ and CDF $\Phi_{\mu,\sigma^2}(x)$ for the general normal distribution with mean value μ and standard deviation σ are given by the formulas

$$\varphi_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \tag{2.8}$$

$$\Phi_{\mu,\sigma^2}(x) = \int_{-\infty}^{x} \varphi_{\mu,\sigma^2}(x)\mathrm{d}x = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)\mathrm{d}x \tag{2.9}$$

In the case where both $\underline{r}$ and $\underline{s}$ are independent random variables following normal distributions, the corresponding performance function $\underline{g} = \underline{r} - \underline{s}$ follows also a normal distribution with the following mean and standard deviation:

$$\mu_g = \mu_r - \mu_s \tag{2.10}$$

$$\sigma_g = \sqrt{{\sigma_r}^2 + {\sigma_s}^2} \tag{2.11}$$

As a result, the PDF and the CDF for the performance function of Eq. (2.5) can be calculated analytically by

$$f_g(x) = \varphi_{\mu_g,{\sigma_g}^2}(x) \tag{2.12}$$

$$F_g(x) = \int_{-\infty}^{x} f_g(x)\mathrm{d}x \tag{2.13}$$

The value of the CDF $F_g(x)$ is the area of the PDF $f_g(x)$ for the region (-∞, x], equal to the probability of $\underline{g} = g(\underline{x})$ being less than x. Thus, the probability of failure $g(\underline{x}) \le 0$ can be calculated as

$$p_\mathrm{f}(\underline{x}) = F_g(0) = \int_{-\infty}^{0} f_g(x)\mathrm{d}x \tag{2.14}$$

By transforming the PDF and CDF of the normal distribution of $\underline{g} = g(\underline{x})$ into the corresponding ones of the standard normal distribution of Eqs. (2.6) and (2.7) we obtain

$$f_g(0) = \varphi(-\frac{\mu_g}{\sigma_g}) \tag{2.15}$$

$$p_\mathrm{f}(\underline{x}) = F_g(0) = \Phi(-\frac{\mu_g}{\sigma_g}) = \Phi(-\beta) \tag{2.16}$$

where $$\beta = \frac{\mu_g}{\sigma_g} \tag{2.17}$$

The parameter β is called *reliability index* and measures the distance between the mean value of the performance function and the limit state surface, in standard deviation units. Figure 2.2 shows a graphical interpretation of the reliability index.

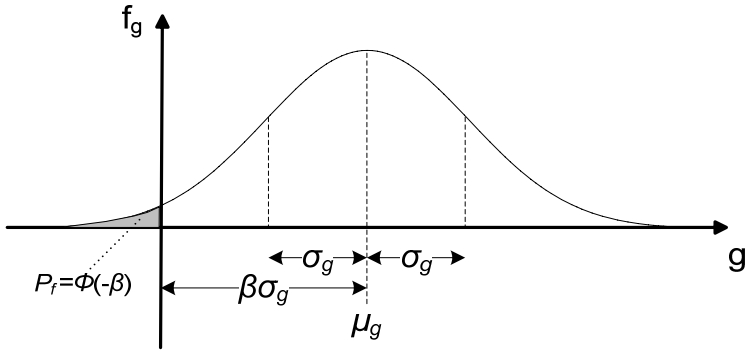


Figure 2.2 Graphical interpretation of the reliability index.

In general, the higher the reliability index the lower the probability of failure which means that the structural reliability and safety are improved. Figure 2.3 depicts the relationship between the probability of failure and the reliability index.

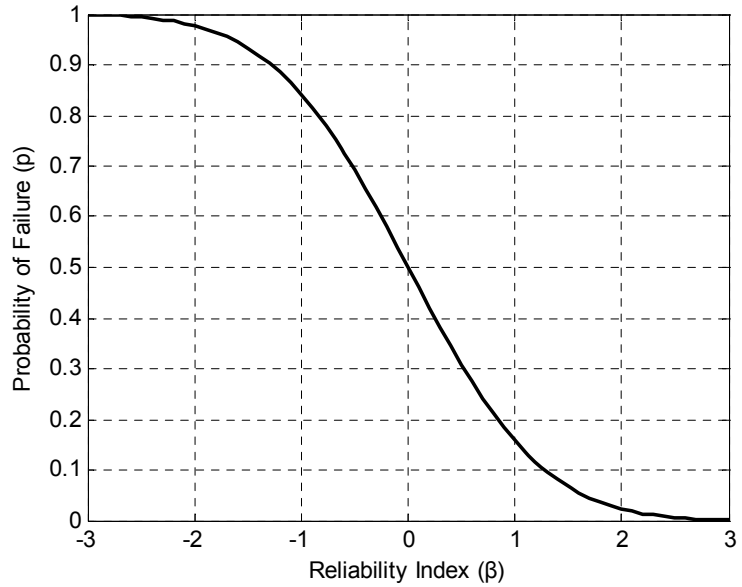


Figure 2.3 Probability of failure *p* as a function of the reliability index *β*.

Figure 2.4 shows a numerical example of the above case, where μ_r=10, σ_r=4 and μ_s=2, σ_s=3. The PDFs for capacity and demand are drawn, as well as the PDF for the corresponding performance function that can be calculated from Eqs. (2.10) and (2.11), resulting

in μ_g=8 and σ_g=5. The grayed region of the PDF of the performance function corresponds to the failure domain where $g \leq 0$, while its area is equal to the probability of failure, with a value of p_f=0.0548 for this specific example. The reliability index, which can be calculated by Eq. (2.17), is β=1.6.

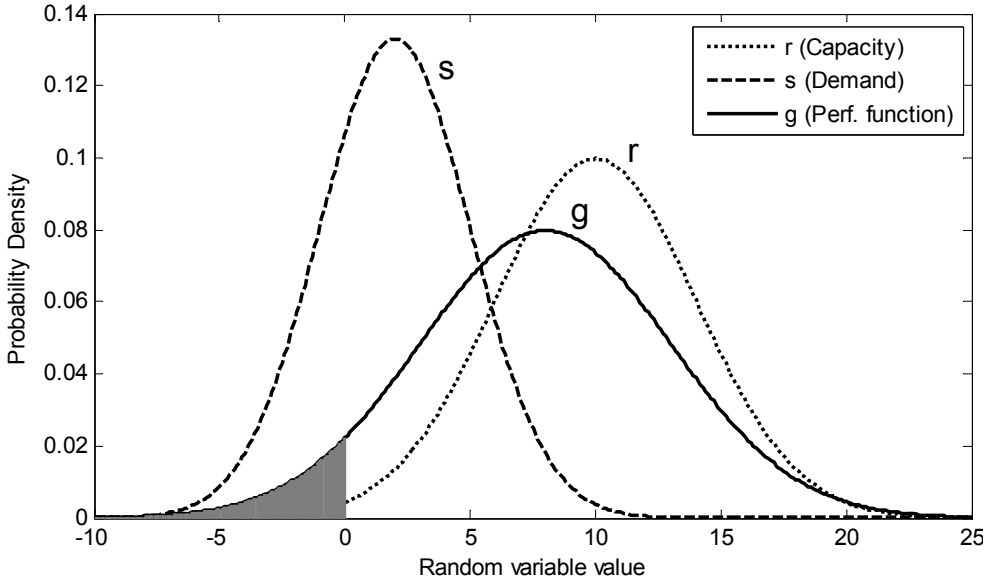


Figure 2.4 PDFs for capacity, demand and the corresponding performance function for the numerical example.

The joint PDF for the specific case of the two normal random variables $\underline{r}$ and $\underline{s}$ can also be calculated analytically. The general formula for the joint PDF of the *multivariate normal distribution* $\varphi_{\boldsymbol{x}}(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}$ of a random vector $\underline{\boldsymbol{x}} = [\underline{x}_1, ..., \underline{x}_m]^{\mathrm{T}}$, is given by

$$\varphi_{\boldsymbol{x}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{m/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right) \tag{2.18}$$

with

$$\varphi_{\boldsymbol{x}}(\boldsymbol{x}) > 0, \quad \int_{\mathbb{R}^m} \varphi_{\boldsymbol{x}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1 \tag{2.19}$$

where $\boldsymbol{x} = [x_1, ..., x_m]^{\mathrm{T}}$ and the vector $\boldsymbol{\mu} = [\mu_1, ..., \mu_m]^{\mathrm{T}}$ contains the mean values (or *expected values*) of each random variable. The expected value of a random variable is denoted with the operator $\mathbb{E}(\cdot)$. Thus,

$$\mu_i = \mathbb{E}(\underline{x}_i) \tag{2.20}$$

while $\boldsymbol{\Sigma}$ is a non-singular covariance matrix, a matrix of covariances between the elements of the random vector $\underline{\boldsymbol{x}}$. The covariance matrix is the natural generalization to higher dimensions of the concept of the variance of a scalar-valued random variable. Each entry of the covariance matrix is the covariance

$$\Sigma_{i,j} = \mathrm{cov}(\underline{x}_i, \underline{x}_j) = \mathbb{E}\left((\underline{x}_i - \mu_i)(\underline{x}_j - \mu_j) \right) \tag{2.21}$$

Thus
$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathrm{cov}(\underline{x}_1, \underline{x}_1) & \mathrm{cov}(\underline{x}_1, \underline{x}_2) & \cdots & \mathrm{cov}(\underline{x}_1, \underline{x}_m) \\ \mathrm{cov}(\underline{x}_2, \underline{x}_1) & \mathrm{cov}(\underline{x}_2, \underline{x}_2) & \cdots & \mathrm{cov}(\underline{x}_2, \underline{x}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{cov}(\underline{x}_m, \underline{x}_1) & \mathrm{cov}(\underline{x}_m, \underline{x}_2) & \cdots & \mathrm{cov}(\underline{x}_m, \underline{x}_m) \end{bmatrix} \tag{2.22}$$

In the case of two random variables $\underline{x}$ and $\underline{y}$, the joint PDF for the bivariate normal distribution is given by

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{z}{2(1-\rho^2)}\right) \tag{2.23}$$

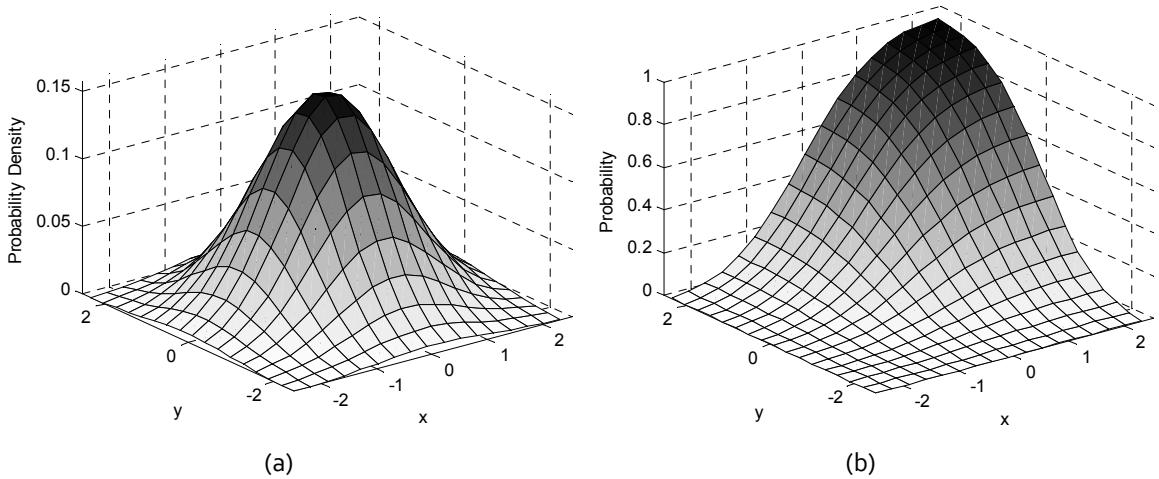
with
$$f(x,y) > 0, \quad \int_{\mathbb{R}^2} f(x,y)\mathrm{d}x\mathrm{d}y = 1 \tag{2.24}$$

where
$$z = \frac{(x-\mu_x)^2}{{\sigma_x}^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{{\sigma_y}^2} \tag{2.25}$$

and ρ is the correlation between $\underline{x}$ and $\underline{y}$. In this case, the 2×2 covariance matrix is given by

$$\boldsymbol{\Sigma} = \begin{bmatrix} {\sigma_x}^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & {\sigma_y}^2 \end{bmatrix} \tag{2.26}$$

In the multivariate case, if the m random variables are independent, there is no correlation between them and the covariance matrix is diagonal. For independent standard normal random variables, the covariance matrix is the identity matrix $\boldsymbol{I}$.



(a) (b)

Figure 2.5 Standard bivariate normal distribution with no correlation ($\rho = 0$): (a) PDF and (b) CDF.

Figure 2.5 shows the PDF and the CDF for the bivariate standard normal distribution with no correlation between the two random variables (ρ=0), plotted for a distance 2.5 times the standard deviation from the mean value for each random variable.

In our case, with also no correlation between capacity and demand (ρ=0), the bivariate joint PDF $f(r,s)$ is given by

$$f(r,s) = \frac{1}{2\pi\sigma_r\sigma_s}\exp\left(-\frac{\dfrac{(r-\mu_r)^2}{\sigma_r{}^2}+\dfrac{(s-\mu_s)^2}{\sigma_s{}^2}}{2}\right) \tag{2.27}$$

The bivariate joint PDF of the two random variables $\underline{r}$ and $\underline{s}$ is plotted in Figure 2.6 as a surface in 3D, the vertical axis denoting the probability density. The limit state surface, defined by the plane $r=s$ is also plotted, which cuts the joint PDF surface dividing it into two regions, the lower left being the failure region (where $r \le s$) and the upper right being the safety region (where $r>s$). The volume of the whole joint PDF is unity, as shown in Eq. (2.24), while the volume of the failure region is equal to the probability of failure (0.0548).

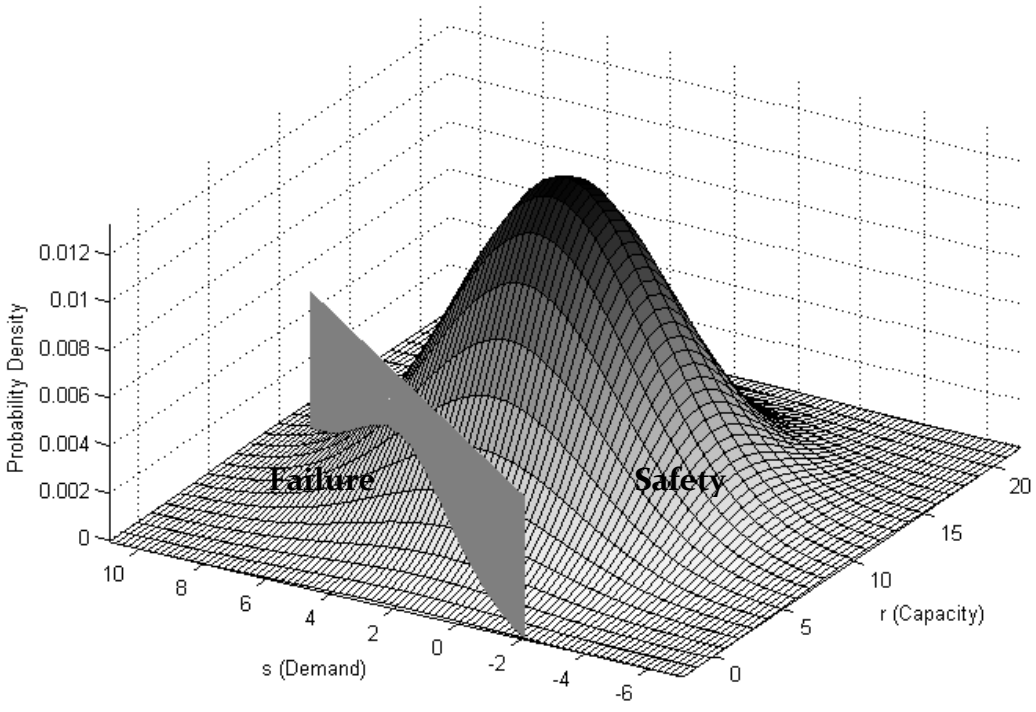


Figure 2.6 Joint PDF for capacity and demand cut by the limit state surface (plane $r=s$).

Figure 2.7 depicts the same diagram in a contour plot, where the limit state surface is plotted as the line $r=s$.

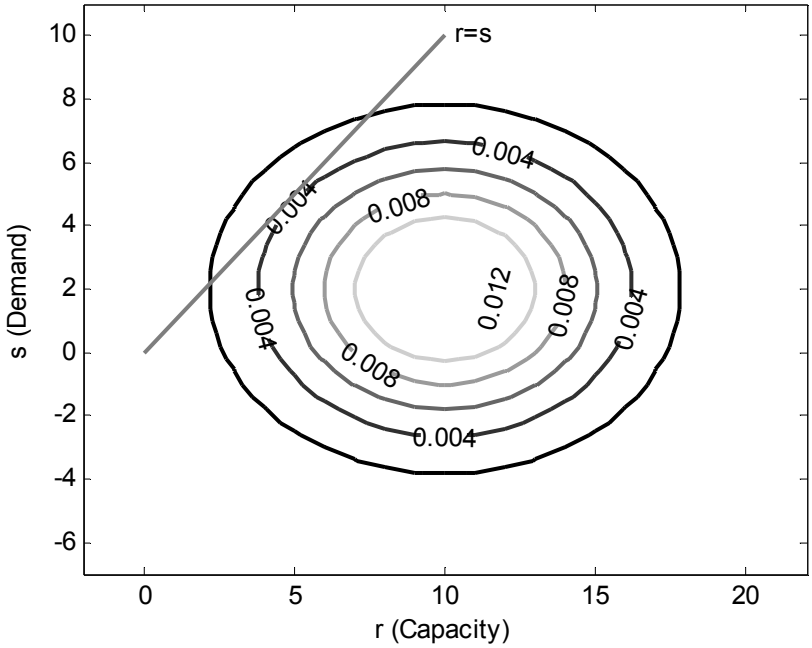


Figure 2.7 Contour plot of the joint PDF for capacity and demand cut by the limit state surface (line *r*=*s*).

In the above example, the PDF of the performance function and the probability of failure can be calculated analytically using well known formulas for the normal distribution. However, analytical methods tend to be applicable only to special kinds of uncertainty distributions and rather simple problems only. In practice, the performance function cannot be written in a simple linear form of normal random variables and it is thus necessary to evaluate the failure probability by calculating the general integral of Eq. (2.3). Direct integration is practically impossible even for small structures, due to the computational cost of the integration in the multi-dimensional space. Numerical methods have to be applied to give an approximation of the failure probability. Three types of numerical methods are commonly used for this purpose:

i. *First- and Second-Order Reliability Method* (FORM/SORM);

ii. *Response Surface Method* (RSM);

iii. *Monte Carlo Simulation* (MCS) method.

2.4 First- and Second-Order Reliability Methods (FORM/SORM)

The First-Order Reliability Method (Hasofer and Lind 1974; Rackwitz and Fiessler 1978) and the Second-Order Reliability Method (Breitung 1984; Der Kiureghian et al. 1987; Fiessler et al. 1979; Köylüoglu and Nielsen 1994) are based on the approximation of the performance function in the standard Gaussian space by using polynomial series. The purpose is to get an approximation of the failure probability. The failure surface in the space of the standard normal variables is approximated at the point on the failure surface

where the probability density of the normalized variables is the highest. The reason for choosing that particular point is that the failure surface should be best approximated in the area which contributes most to the integral defining the probability of failure. Because of the symmetry of the distribution of the variables in the standard normal space, this design point, called also the *Most Probable Failure Point* or simpler *Most Probable Point* (MPP) or *β-point*, is the nearest failure point to the origin having the highest probability density among all points in the failure domain, in the standard normal space. First and second order approximate reliability methods entail prior knowledge of the mean and the variance of each random variable, while a differentiable failure function is also required.

The basic steps in order to implement FORM/SORM are the following (Rodriguez and Montero 2003):

1. Transformation of the basic variables into standard and uncorrelated normal variables (in the so-called standard normal space). As a result, the real joint probability density function is transformed into an "equivalent" multivariate normal density (with zero mean values and identity covariance matrix).
2. Determination of the MPP (the design point) in the standard normal space.
3. Approximation of the limit state surface in the standard normal space at the design point with the FORM or SORM principle.
4. Computation of the probability of failure in accordance with the approximation surface selected in *step 3*.

In order to search for the design point, an optimization algorithm should be applied to the following optimization problem:

$$\boldsymbol{u}^* = \min\{\|\boldsymbol{u}\| \quad | G(\boldsymbol{u}) = 0\} \tag{2.28}$$

where $\boldsymbol{u}^* = [u_1^*,...,u_m^*]^{\mathrm{T}}$ is the design point and $G(\boldsymbol{u})$ is the limit state function in the transformed standard normal space, where $G(\boldsymbol{u}) \leq 0$ denotes failure.

The presence of various local optima for the optimization problem of Eq. (2.28) can cause significant problems to FORM and SORM (Der Kiureghian and Dakessian 1998). Firstly, if the optimization algorithm converges to a local sub-optimum rather than the global design point, the FORM and SORM solutions will miss the region of dominant contribution to the failure probability integral and hence the corresponding approximation will be in gross error. Secondly, even if the optimization algorithm converges to the global design point, there could be significant contributions to the failure probability integral from the neighborhoods of the local design points and as a result approximating the limit-state surface only at the global design point will not account for these contributions.

2.4.1 FORM principle

FORM (Hasofer and Lind 1974; Rackwitz and Fiessler 1978) has been used extensively by engineers for nearly two decades. In FORM, the failure surface in the space of the standard normal variables is linearized (as a hyperplane) at the point on the failure surface where the probability density of the normalized variables is highest.

The first-order approximation of G at the design point is given by the first-order Taylor series expansion as

$$G(\boldsymbol{u}) \simeq G(\boldsymbol{u}^*) + \nabla_u G(\boldsymbol{u}^*)^{\mathrm{T}} \left(\boldsymbol{u} - \boldsymbol{u}^* \right) \tag{2.29}$$

where $\nabla_u G(\boldsymbol{u}^*)$ is the gradient of G at the design point $\boldsymbol{u}^*$. Figure 2.8 shows a graphical interpretation of the FORM principle in the standard normal space for a two dimensional case.

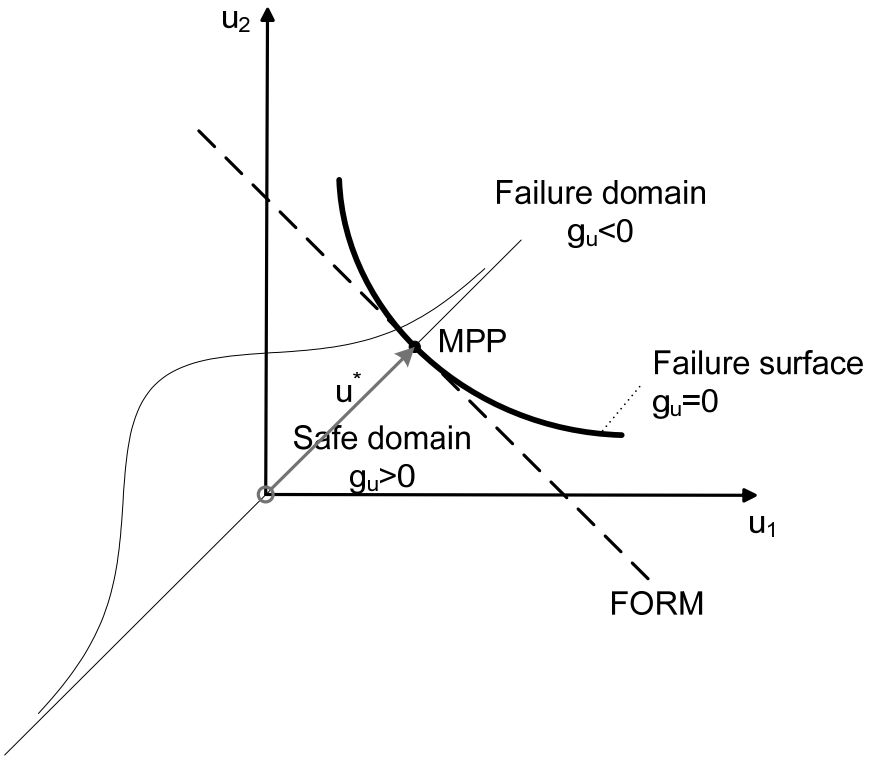


Figure 2.8 Graphical interpretation of the FORM principle.

The corresponding reliability index and the probability of failure for FORM are given by

$$\beta = sign(\, g_u(\mathbf{0}))\left\| \boldsymbol{u}^* \right\| \tag{2.30}$$

$$p_{\mathrm{FORM}} = \Phi(-\beta) \tag{2.31}$$

2.4.2 SORM principle

In SORM (Breitung 1984; Der Kiureghian et al. 1987; Fiessler et al. 1979; Köylüoglu and Nielsen 1994), the failure surface in the space of the standard normal variables is approximated by a quadratic function (parabolic surface) at the point on the failure surface where the probability density of the normalized variables is highest. The second-order approximation of *G* at the design point is given by the second-order Taylor series expansion as

$$G(\boldsymbol{u}) \simeq G(\boldsymbol{u}^*) + \nabla_u G(\boldsymbol{u}^*)^{\mathrm{T}} \left(\boldsymbol{u} - \boldsymbol{u}^* \right) + \frac{1}{2} \left(\boldsymbol{u} - \boldsymbol{u}^* \right)^{\mathrm{T}} \nabla_u^2 G(\boldsymbol{u}^*) \left(\boldsymbol{u} - \boldsymbol{u}^* \right) \qquad (2.32)$$

where $\nabla_u G(\boldsymbol{u}^*)$ is the gradient of *G* and $\nabla_u^2 G(\boldsymbol{u}^*)$ is the Hessian matrix of G at the design point $\boldsymbol{u}^*$. Figure 2.9 shows a graphical interpretation of the SORM principle in the standard normal space for a two dimensional case. It can be clearly seen that the second-order approximation by SORM incorporates better the influence of the curvature of the limit state surface at the design point, as compared to the FORM case of Figure 2.8.

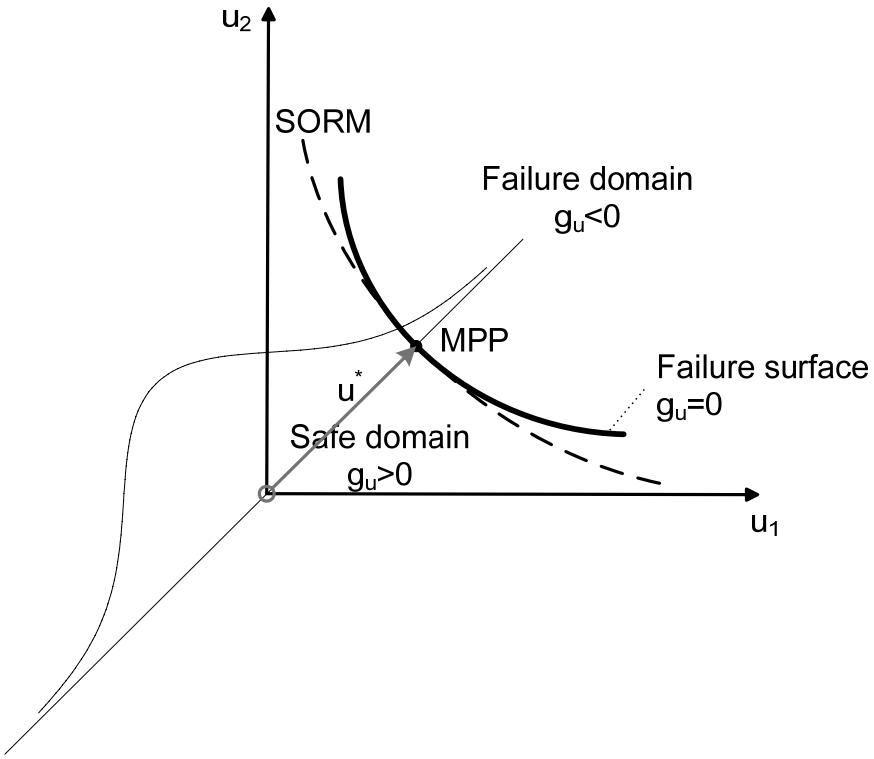


Figure 2.9 Graphical interpretation of the SORM principle.

Second-order integration involves applying a curvature correction for the calculation of the probability of failure. Two simple approximations, one given by Breitung (1984) and the other one given by Hohenbichler and Rackwitz (1988) can be used to obtain the second-order reliability estimates. Breitung's approximation is given by

$$p_{\mathrm{SORM1}} = \Phi(-\beta)\prod_{i=1}^{m-1}\frac{1}{\sqrt{(1+2\lambda_i\beta)}} \tag{2.33}$$

where λ_i=-κ_i/2; and κ_i are the main curvatures, taken positive for a concave limit state function. Hohenbichler and Rackwitz's approximation is given by

$$p_{\mathrm{SORM2}} = \Phi(-\beta)\prod_{i=1}^{m-1}\frac{1}{\sqrt{(1+2n(\beta)\lambda_i\beta)}} \tag{2.34}$$

where $n(\beta)=\varphi(\beta)/\Phi(-\beta)$. Both formulas perform well for moderate to large values of β and approach the FORM results as β tends to zero (Hong 1999). The corresponding reliability index for SORM is be given by

$$\beta_{\mathrm{SORM}} = -\Phi^{-1}(p_{\mathrm{SORM}}) \tag{2.35}$$

where p_{SORM} can be either p_{SORM1} or p_{SORM2}, depending on the method used.

2.5 Response Surface Method

RSM tries to approximate the mechanical response of the structure by using the so-called metamodel. Suppose $\underline{\boldsymbol{x}} = [\underline{x}_1,...,\underline{x}_m]^{\mathrm{T}}$ is a vector of m random variables and $\underline{g} = g(\underline{\boldsymbol{x}})$ is the corresponding performance function. Although the performance function and the actual response in general are functions of the random variables, these functions are generally unavailable in closed form for structural reliability problems. In RSM, "experiments" are conducted with the random variables for a sufficient number of times in order to define the response surface to the level of accuracy desired. Each experiment can be represented as a "point" in the m-dimensional space of the random variables. For each point, a structural analysis is performed and a value of the performance function g is calculated. The basic response surface procedure aims at approximating the performance function with a polynomial $\tilde{g}(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}$. The unknown coefficients of the polynomial are determined, such that the error of the approximation is minimum in the region of interest.

The selection of the order of the approximating polynomial and of the points in $\mathbb{R}^m$ for experimentation is of great importance, requiring careful consideration. The degree of $\tilde{g}(\boldsymbol{x})$ should be less than or equal to the degree of $g(\underline{\boldsymbol{x}})$ to get a well-conditioned system of linear equations for the unknown coefficients (Rajashekhar and Ellingwood 1993). Of course, the function $g(\underline{\boldsymbol{x}})$ itself is not known a priori. If $\tilde{g}(\boldsymbol{x})$ is of much higher degree than $g(\underline{\boldsymbol{x}})$, one obtains an ill-conditioned system of equations. Moreover, higher order polynomials can exhibit erratic behavior in the sub-domains not covered by the experiments. Up to a certain degree, a higher order polynomial improves the accuracy of the approximation at the expense of additional computational time. The rate of increase in accuracy reduces with the degree of the polynomial but the computational cost increases exponentially, as a higher order polynomial involves greater number of unknown coeffi-

cients and requires correspondingly more structural analyses. For reliability estimates, one needs to have a good approximation of the performance function around the design point, or the region of the failure domain where the joint probability density is relatively large and thus contributes most to the overall failure probability. Since the actual limit state function and the actual design point are not known, the accuracy of the reliability estimate depends on the accuracy of the polynomial approximation in the region of the design point. Quadratic polynomials have shown to be suitable for localized approximation of structural systems in general.

2.5.1 Advantages and disadvantages of RSM for reliability analysis

The main advantages of RSM (Chateauneuf 2008) are (i) the reduction of the computational cost for moderate number of random variables; and (ii) (for reliability-based design optimization), the possibility of coupling reliability and optimization algorithms to achieve high efficiency. The most common drawback lies in the large number of Finite Element analyses of a probably complex model, for moderate and high number of random variables. It should be noted that the large part of the computational cost lies in the evaluation of the polynomial coefficients. After the polynomial has been defined, the failure probability can be simply evaluated by using the response surface which is an easy to calculate analytical expression.

2.6 Monte Carlo Simulation

MCS methods are a class of computational algorithms that rely on repeated random sampling to compute their results and are often used for simulating physical and mathematical systems. They are mainly used for obtaining numerical solutions when it is infeasible or impossible to compute an exact result analytically. Because of their reliance on repeated computation and random numbers, they are most suited to calculation by a computer. The term *Monte Carlo method* was coined in the 1940s by physicists working on nuclear weapon projects in the Los Alamos National Laboratory, in reference to games of chance, a popular attraction in the casino of Monte Carlo in Monaco (Metropolis 1987; Metropolis and Ulam 1949).

These methods are especially useful in studying systems with a large number of coupled degrees of freedom, phenomena with significant uncertainty in inputs, or for the evaluation of definite multidimensional integrals with complicated boundary conditions. They are used to solve various problems by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties.

MCS can be used for analyzing uncertainty propagation, where the goal is to determine how random variation, lack of knowledge, or error affects the sensitivity, performance, or reliability of the system that is being modeled. It is categorized as a sampling method because the inputs are randomly generated from probability distributions to simulate the process of sampling from an actual population. The choice of distribution for the inputs

should most closely match the available data, or best represent the current state of knowledge. MCS technique has the important property that the successive points in the sample are independent.

The MCS method is often applied in three fields of application (Nowak and Collins 2000):

i. To solve complex problems for which closed-form solutions are either impossible or extremely difficult.

ii. To solve complex problems that can be solved (at least approximately) in a closed form provided that some simplifying assumptions are made to the original problem. By using MCS the original problem can be studied without any assumptions and thus more realistic results can be obtained.

iii. To check the results of other solution techniques.

2.6.1 Advantages and disadvantages of MCS for reliability analysis

The main advantages of MCS method (Chateauneuf 2008) are: (i) the capability of handling practically any mechanical or physical model regardless of its complexity; and (ii) its simple implementation without any modification of the mechanical model which can be considered as a "black box" receiving simple analysis calls. The main disadvantages are: (i) the excessive computational effort due to the enormous sample size required, especially for realistic structures with low probability of failure; and (ii) the numerical noise due to random sampling, leading to non-monotonic estimates during simulations, and as a result, it becomes impossible to get accurate and stable evaluation of the response gradient. Although the former shortcoming can be alleviated by using variance reduction techniques, as will be discussed in detail in Section 2.7, the latter still remains a serious difficulty for practical applications where there is a need for gradient information.

2.6.2 Calculation of basic statistical quantities for one random variable with MCS

Suppose $\underline{x}$ is a real-valued random variable with PDF $f(x)$:

$$f(x) > 0, \quad \int_{-\infty}^{\infty} f(x)\mathrm{d}x = 1 \tag{2.36}$$

Let $g(\underline{x})$ be an arbitrary real function of the random variable $\underline{x}$ and $\underline{g} = g(\underline{x})$ the corresponding random variable. The expected value (mean value or first central moment) and the variance (second central moment) of $\underline{g}$ with respect to the PDF $f(x)$ are given by the analytical formulas:

$$\mu = \mathbb{E}(\underline{g}) = \int_{-\infty}^{\infty} g(x) f(x)\mathrm{d}x \tag{2.37}$$

$$\sigma^2 = \mathrm{var}(\underline{g}) = \mathbb{E}\left(\left(\underline{g} - \mu \right)^2 \right) = \int_{-\infty}^{\infty} \left(g(x) - \mu \right)^2 f(x)\mathrm{d}x \tag{2.38}$$

Making n random drawings of $\underline{x}$ $(x_1, \ldots, x_n)$, called *simulation runs*, the corresponding values $g(x_1), \ldots, g(x_n)$ for the sample can be calculated and their mean value is given by:

$$\mu_n(\underline{g}) = \frac{1}{n} \sum_{i=1}^{n} g(x_i) \tag{2.39}$$

For a given number of simulation runs n, the quantity $\mu_n(\underline{g})$ represents the simulated value or the *Monte Carlo estimator* of μ. The unbiased sample variance can also be calculated as follows

$$\sigma_n{}^2(\underline{g}) = \frac{1}{n-1} \sum_{i=1}^{n} \left(g(x_i) - \mu_n(\underline{g}) \right)^2 \tag{2.40}$$

The quantity $\sigma_n{}^2(\underline{g})$ represents the Monte Carlo estimator of σ^2. It can be proved that as the number of simulation runs n increases, the calculated mean of the sample converges to the real expected value of $\underline{g}$, while the calculated sample variance converges to the real variance of $\underline{g}$:

$$\lim_{n \to \infty} \mu_n(\underline{g}) = \mu \tag{2.41}$$

$$\lim_{n \to \infty} \sigma_n{}^2(\underline{g}) = \sigma^2 \tag{2.42}$$

2.6.3 Calculation of the probability of failure with MCS

One random variable case

In the previous example, let the arbitrary real function $\underline{g} = g(\underline{x})$ be the limit state function, i.e. $g(\underline{x}) \leq 0$ denotes failure state of the model. The probability of failure is given analytically by the integral expression

$$p_\mathrm{f}(\underline{x}) = \int_{\mathcal{D}_\mathrm{f}} f(x)\mathrm{d}x \tag{2.43}$$

Where $\mathcal{D}_\mathrm{f}$ is the "failure" domain in $\mathbb{R}$, defined as

$$\mathcal{D}_\mathrm{f} = \{ x \in \mathbb{R} \mid g(x) \leq 0 \} \tag{2.44}$$

Making n random drawings of $\underline{x}$ $(x_1, \ldots, x_n)$, the corresponding values $g(x_1), \ldots, g(x_n)$ for the sample can be obtained. A sequence is defined as follows:

$$a_i = \begin{cases} 0 & \text{if} \quad g(x_i) > 0 \\ 1 & \text{if} \quad g(x_i) \leq 0 \end{cases}, \quad \text{for } i = 1, \ldots, n \tag{2.45}$$

The sum of the elements of the sequence counts the samples for which failure has occurred, out of *n* samples in total. The corresponding rate of occurrence can be calculated by

$$p_n(\underline{x}) = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{2.46}$$

For a given number of *n* simulation runs, the quantity $p_n(\underline{x})$ represents the Monte Carlo estimator of $p(\underline{x})$. It can be proved that as the number of simulation runs *n* increases, the Monte Carlo estimator of the probability of violation converges to the real value of $p(\underline{x})$.

$$\lim_{n\to\infty} p_n(\underline{x}) = p(\underline{x}) \tag{2.47}$$

Multiple random variables case

Suppose $\underline{\boldsymbol{x}} = [\underline{x}_1,...,\underline{x}_m]^{\mathrm{T}}$ is a real-valued vector of *m* random variables (structural parameters) with joint PDF $f(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}$

$$f(\boldsymbol{x}) > 0, \quad \int_{\mathbb{R}^m} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1 \tag{2.48}$$

It should be noted that in the general case the individual random variables $\underline{x}_1,...,\underline{x}_m$ can be either correlated with each other or not correlated at all (independent). In any case, the joint PDF $f(\boldsymbol{x})$ contains all the information regarding the random variables' distributions. Let $\underline{g} = g(\underline{\boldsymbol{x}})$ be a real function of the random vector $\underline{\boldsymbol{x}}$ that expresses the limit state function, i.e. $g(\underline{\boldsymbol{x}}) \le 0$ denotes failure of the model. The probability of failure is given analytically by the integral expression

$$p_{\mathrm{f}}(\underline{\boldsymbol{x}}) = \int_{\mathcal{D}_{\mathrm{f}}} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \tag{2.49}$$

Where $\mathcal{D}_{\mathrm{f}}$ is the "failure" domain in $\mathbb{R}^m$, defined as

$$\mathcal{D}_{\mathrm{f}} = \{\boldsymbol{x} \in \mathbb{R}^m \mid g(\boldsymbol{x}) \le 0\} \tag{2.50}$$

Making *n* random drawings of $\underline{\boldsymbol{x}}$ ($\boldsymbol{x}^1,...,\boldsymbol{x}^n$), the corresponding values $g(\boldsymbol{x}^1),...,g(\boldsymbol{x}^n)$ for the sample can be calculated. It should be noted that each random drawing of $\underline{\boldsymbol{x}}$ (sample) is in fact a vector $\boldsymbol{x}^i = [x_1^i,...,x_m^i]^{\mathrm{T}}$ containing the values of the of *m* random variables. A sequence is defined as follows:

$$a_i = \begin{cases} 0 & \text{if} \quad g(\boldsymbol{x}^i) > 0 \\ 1 & \text{if} \quad g(\boldsymbol{x}^i) \le 0 \end{cases}, \quad \text{for } i = 1,...,n \tag{2.51}$$

The sum of the sequence counts the samples for which the limit state has been exceeded, out of *n* samples in total. The corresponding rate of occurrence can be calculated by

$$p_n(\underline{\boldsymbol{x}}) = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{2.52}$$

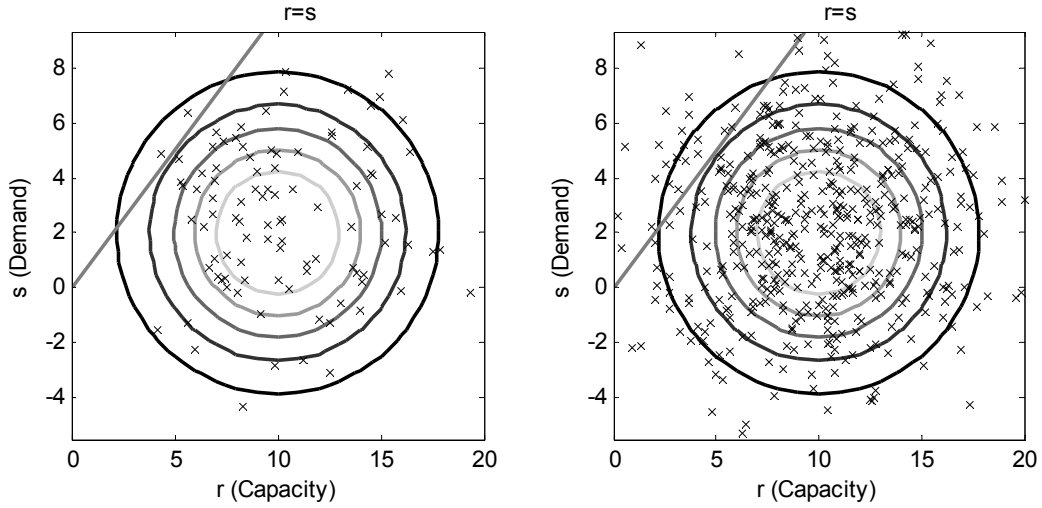
For a given number of n simulation runs, the quantity $p_n(\underline{\boldsymbol{x}})$ represents the Monte Carlo estimator of $p(\underline{\boldsymbol{x}})$. It can be proved that as the number of simulation runs n increases, the Monte Carlo estimator of the probability of violation converges to the real value of $p(\underline{\boldsymbol{x}})$.

$$\lim_{n\to\infty} p_n(\underline{\boldsymbol{x}}) = p(\underline{\boldsymbol{x}}) \tag{2.53}$$

Numerical example

Suppose that the probability of failure for the two-variable problem of Figure 2.7 is to be calculated with MCS. The analytical "exact" solution, as was shown in Section 2.3.3, is p_f=0.0548. Figure 2.10 depicts the joint PDF of the two random variables in a contour plot, together with the representation of the MCS samples. Each sample is depicted as an "×" in the two-dimensional space of the figure. The picture on the left depicts 100 MCS samples, while the picture on the right depicts 500 MCS samples. The corresponding probabilities of failures obtained for the two cases are p_{100}=0.03 and p_{500}=0.056. By using a number of 5000 MCS samples, we can obtain p_f=0.0552, a very good approximation of the "true" probability of failure.

Of course, as random numbers are used for the generation of samples, different results will be obtained for other MCS runs, even if the same number of samples is used, as will be shown in detail in Section 2.6.4. In any case, in order to estimate $p(\underline{\boldsymbol{x}})$ with accuracy, an adequate number of n independent random samples should be produced.



(a) (b)

Figure 2.10 MCS for the calculation of probability of failure ("exact" value p_f = 0.0548) (a) p_{100}=0.03 for 100 samples, (b) p_{500}=0.056 for 500 samples.

2.6.4 Accuracy of probability estimates with MCS

The MCS method can provide probability estimates for structural reliability problems. In general, the estimate improves as the number of simulation runs increases. If a MCS for obtaining the probability of failure is repeated for a number of times, different results will be obtained. The calculated value will vary from sample to sample. This means that the probability estimate of the MCS is a random variable itself, with its own mean, standard deviation and coefficient of variation.

Let p_{true} be the theoretically correct probability that is tried to be estimated by MCS. It can be shown (Soong and Grigoriou 1993) that the expected value, variance and coefficient of variation of the estimated probability $\underline{p}$ are given by the formulas:

$$\mathbb{E}(\underline{p}) = p_{\text{true}} \tag{2.54}$$

$$\sigma_p^2 = \frac{1}{n}\left(p_{\text{true}}(1 - p_{\text{true}}) \right) \tag{2.55}$$

$$v_p = \frac{\sigma_p}{\mathbb{E}(\underline{p})} = \sqrt{\frac{1 - p_{\text{true}}}{n \cdot p_{\text{true}}}} \tag{2.56}$$

It is clear from Eq. (2.55) that the uncertainty in the estimate of the probability decreases as the total number of simulation runs, n, increases, as expected. The above formulas provide a way to determine how many simulations are required to estimate a probability and limit the uncertainty in the estimate. The number of simulations needed to obtain a given probability of failure, while keeping the coefficient of variation at a certain value, can be obtained by Eq. (2.56) as follows:

$$n = \frac{1 - p_{\text{true}}}{v_p^2 \cdot p_{\text{true}}} \tag{2.57}$$

As an example, if one intends to estimate a probability as low as 6×10^{-3} and keep the coefficient of variation at or below 20%, a sample size $n = 4142$ is needed.

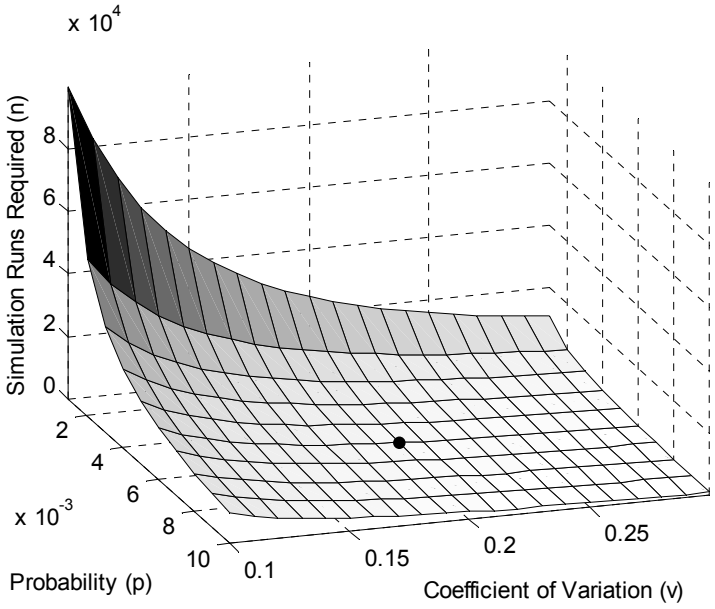


Figure 2.11 Required simulation runs *n* as a function of the probability of failure *p* and the coefficient of variation *v*.

In Figure 2.11 the required number of simulation runs is plotted as a function of the probability of failure and the coefficient of variation. The dot in the figure represents the above mentioned numerical example that corresponds to point (6×10^{-3}, 0.20, 4142) in the 3D space of the figure. It can be seen that as the probability and the coefficient of variation decreases, the required number of simulation runs increases excessively.

2.7 Improved sampling techniques

In order to improve the accuracy of the MCS estimate or, in other words, reduce the estimate's variance, a common, obvious solution is to increase *n*, as the estimate's variance is inversely proportional to *n* as shown in Eq. (2.56). This method has the disadvantage of requiring more calculations as the sample size increases. As a result, for typical structural reliability problems the computational effort involved in the basic MSC can become excessive due to the enormous sample size required.

An alternative means to reduce the computational effort of MCS is by using variance reduction techniques that use statistical approaches which obtain more information from the computer runs conducted, or control and direct the pseudo-random streams to optimize the information to be produced by a run (James 1985). Various variance reduction techniques have been proposed. Some examples are:

1. *Importance Sampling* (IS) (Anderson 1999; Melchers 1989; Song 1997; Srinivasan 2002);
2. *Latin Hypercube Sampling* (LHS) (Florian 1992; McKay et al. 1979);
3. *Descriptive Sampling* (DS) (Saliby 1990; Saliby 1997);

4. Control Variates (L'Ecuyer and Buist 2008; Szechtman 2003);
5. Antithetic Variates (Fishman and Huang 1983; Ross 2006);
6. Adaptive Sampling (Bucher 1988; Mori and Ellingwood 1993; Thompson and Seber 1996);
7. *Hammersley Sequence Sampling* (HSS) (Kalagnanam and Diwekar 1997; Wang et al. 2004);
8. Line Sampling (Koutsourelakis et al. 2004);
9. Subset Simulation (Au and Beck 2001);
10. Directional Simulation (Gray and Melchers 2006; Nie and Ellingwood 2004; Nie and Ellingwood 2005).

Recent results (Koutsourelakis et al. 2004) reveal that variance reduction techniques still require significant number of the system response evaluations to estimate failure probabilities of the order less than 10^{-3}. In this thesis, three sampling methodologies are mainly used: (i) The Crude MCS; (ii) Importance Sampling; and (iii) MCS with Latin Hypercube Sampling. IS and MCS with LHS methodologies will be described in detail in Sections 2.8 and 2.9, respectively, while a brief description of other methods is also given in Section 2.10.

2.8 Latin Hypercube Sampling (LHS)

One of the advantages of MCS is the fact that its results can be treated using classical statistical methods, thus results can be presented in the form of histograms and methods of statistical estimation and inference can be applicable (Diwekar 2008). Nevertheless, in most applications, the actual relationship between successive points in a sample has no physical significance; hence the randomness/independence for approximating a distribution is not crucial. Moreover, the error of approximating a distribution by a finite sample depends on the equidistribution properties of the sample rather than its randomness. Once it is apparent that the uniformity properties are central to the design of sampling techniques, constrained or stratified sampling techniques became appealing.

The idea of the *Latin Hypercube Sampling* (LHS) technique was proposed by MacKay et al. (1979) in an effort to reduce the required computational cost of random sampling methodologies. LHS is one form of stratified sampling that can yield more precise estimates of the distribution function. In the context of statistical sampling, a square grid containing sample positions is a Latin square if (and only if) there is only one sample in each row and each column. Figure 2.12 depicts a Latin square, an example of eight samples in a two-dimensional space. Note that every black square, that represents a sample, is unique in its row and column. The generalization of this concept to an arbitrary number of m dimensions constitutes a Latin hypercube of dimension m, where each sample is the only one in each axis-aligned hyperplane containing it.

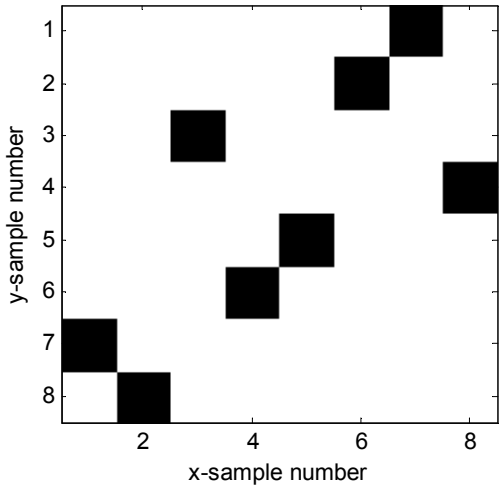


Figure 2.12 Possible random pairing for LHS sampling with two variables (8 samples).

Based on this concept, Latin hypercube samples are generated by dividing the range of each of the *m* uncertain variables into *N* non-overlapping segments of equal probability. Thus, the *m*-dimensional parameter space is partitioned into N^m cells. For each random variable, a single value is selected from each interval at random with respect to the probability distribution in the interval, producing a set of *N* values. In *Median Latin Hypercube Sampling* (MLHS) this value is chosen as the mid-point of the interval. In the case of a random variable $\underline{x}$ with PDF $f(x)$, points $x_1,...,x_{N-1}$ divide its range $(-\infty,+\infty)$ into segments of equal probability $1/N$ as follows

$$\int_{-\infty}^{x_1} f(x)\mathrm{d}x = \int_{x_i}^{x_{i+1}} f(x)\mathrm{d}x = \int_{x_{N-1}}^{+\infty} f(x)\mathrm{d}x = \frac{1}{N}, \quad (i = 1,..., N-2) \tag{2.58}$$

The same is done for every one of the *m* random variables. The values of each random variable are randomly matched with each other to create *N* samples, each one containing *m* random values (*m*-tuplets). The number of intervals *N* needs to be the same for each variable. The LHS method is independent of the random variables number as it does not require more samples for more random variables, which is one of its main advantages.

A schematic representation of the stratification scheme (intervals of equal probability) of LHS for a normal random variable is given in Figure 2.13 in comparison to the crude MCS scheme of Figure 2.14. In Figure 2.13, the total area of the distribution has been divided into $N=8$ regions, each one having an area of $1/8=0.125$. Eight samples have been generated for illustration, each one belonging to each region. It is clear that using the LHS, the distribution of samples is better and more representative of the PDF, than the one using crude MCS.

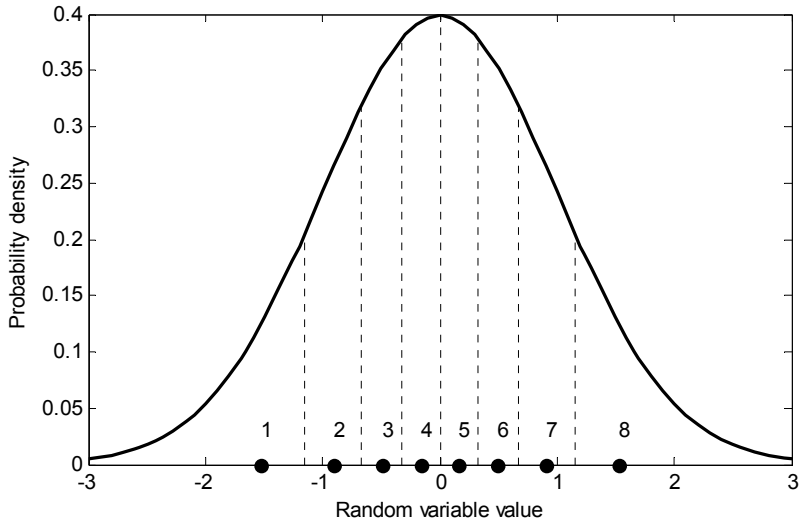


Figure 2.13 Latin Hypercube sampling for the normal distribution with 8 samples.

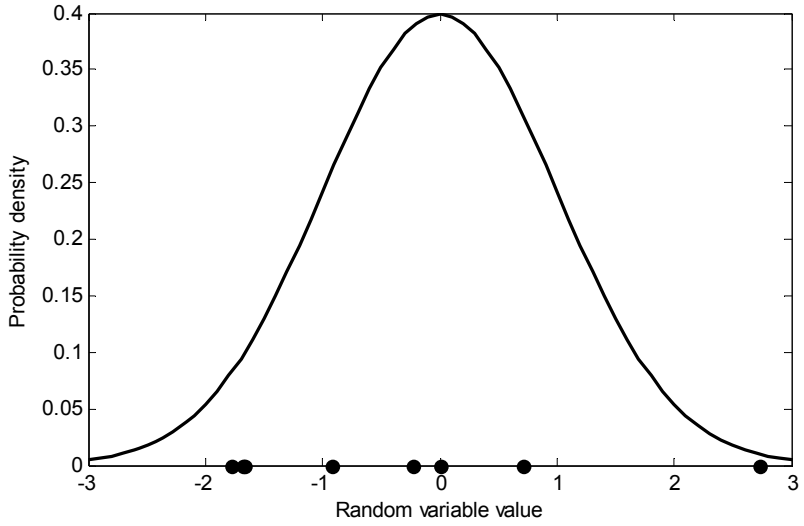


Figure 2.14 MC sampling for the normal distribution with 8 samples.

The advantage of the LHS approach is that the random samples are generated from all ranges of possible values. LHS is generally recognized as one of the most efficient size reduction techniques for the calculation of statistical quantities and relatively large violation probabilities (Owen 1997).

Latin Hypercube sampling can improve the efficiency of MCS by picking the input samples in a more effective way. Whereas MCS method typically pick points at random within the domain, Latin Hypercube samples the entire domain more systematically. It is a strategy for generating random sample points ensuring that all portions of the random space are properly represented. Thus, by means of LHS method, the whole parameter space can be sampled more reliably with fewer samples, improving convergence rates and speeding up the execution time.

It should be noted that in crude MCS, in order to increase the sample size, new random numbers can be generated and they can be added to the existing sample, so existing samples can be taken into account when the sample size is to be increased. On the contrary, in LHS, due to the stratified nature of the method, all samples have to be generated from the beginning. In order to use a larger sample in LHS, one should generate the whole larger sample from the beginning.

2.8.1 Comparison of Crude MCS with MCS-LHS

The performance of the MCS-LHS method compared to the Crude MCS will be examined in two mathematical examples, a simple one-variable problem and a two-variable problem. Both methods are used for the calculation of some statistical quantities and the results are compared.

One-variable problem

Suppose x is a random variable that follows the standard normal distribution. The mean value of x is μ_x=0, while the standard deviation is σ_x=1. Sample sizes starting from 100 samples and ending in 10000 samples with an increment of 100 are used. For a given sample size, random numbers are generated either purely randomly (Crude MCS) or by MCS with LHS (MCS-LHS). Every time, the mean value and the standard deviation are calculated. The figures below depict the calculated values versus the sample size for the mean and the standard deviation.

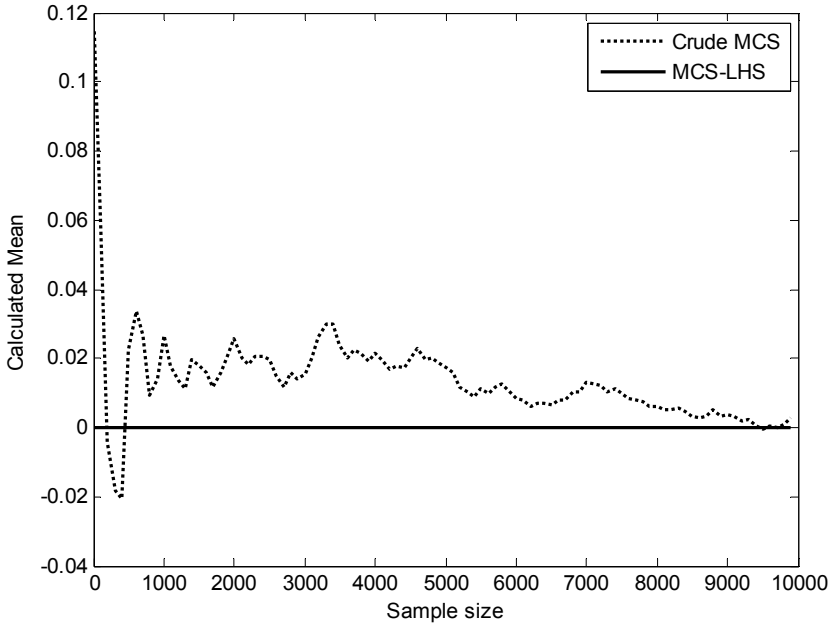


Figure 2.15 MCS-LHS compared to Crude MCS for the calculation of the mean value for the one-variable example.

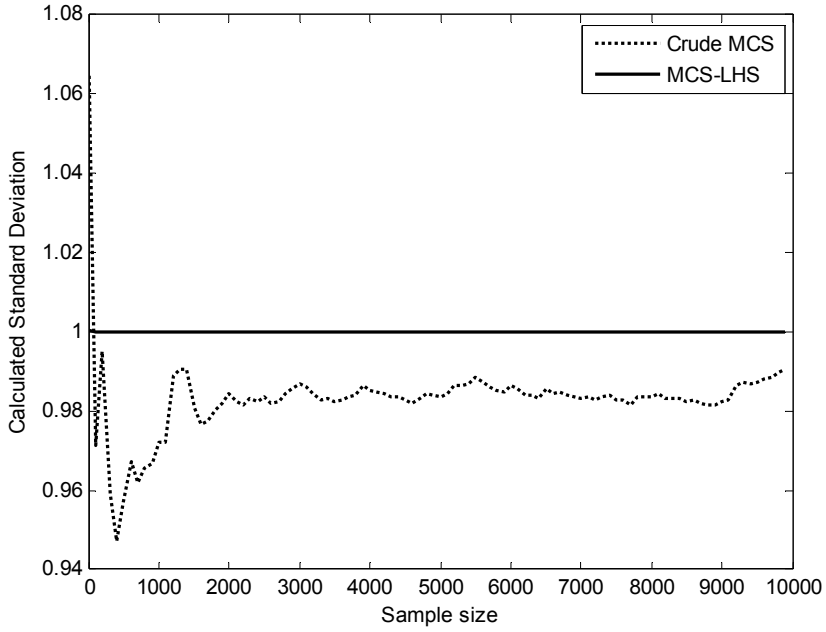


Figure 2.16 MCS-LHS compared to Crude MCS for the calculation of the standard deviation for the one-variable example.

The superiority of MCS-LHS over Crude MCS is clearly demonstrated, as Crude MCS needs too many iterations to converge to the exact values of the mean and sigma, while LHS needs only a few.

Two-variable problem

Suppose $\underline{x}$ and $\underline{y}$ are two real-valued random variables that both follow normal distributions with mean values μ_x, μ_y and standard deviations σ_x, σ_y, respectively. The PDF of the bivariate normal distribution of $\underline{x}$ and $\underline{y}$ is given by Eq. (2.23) for the general case, with possible correlation between the random variables.

In order to examine the performance of the MCS-LHS method compared to the Crude MCS, a numerical test is examined. For the numerical test's purposes, we suppose that the two random variables are independent ($\rho=0$) and that the mean values and standard deviations are the ones given in the table below.

Table 2.1 Statistical parameters of the two independent random variables.

Random variable	Mean	Standard deviation
x	$\mu_x=1$	$\sigma_x=1$
y	$\mu_y=2$	$\sigma_y=3$

For the case with no correlation, the PDF of the bivariate distribution of the two independent normal variables is given by

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-\frac{d}{2}\right) \tag{2.59}$$

with $$f(x,y) > 0, \quad \int_{\mathbb{R}^2} f(x,y)\mathrm{d}x\mathrm{d}y = 1 \tag{2.60}$$

where $$d = \frac{(x-\mu_x)^2}{{\sigma_x}^2} + \frac{(y-\mu_y)^2}{{\sigma_y}^2} \tag{2.61}$$

Figure 2.17 depicts the PDF of the bivariate distribution of the two independent normal variables in the $\mu \pm 2\sigma$ region for both variables.

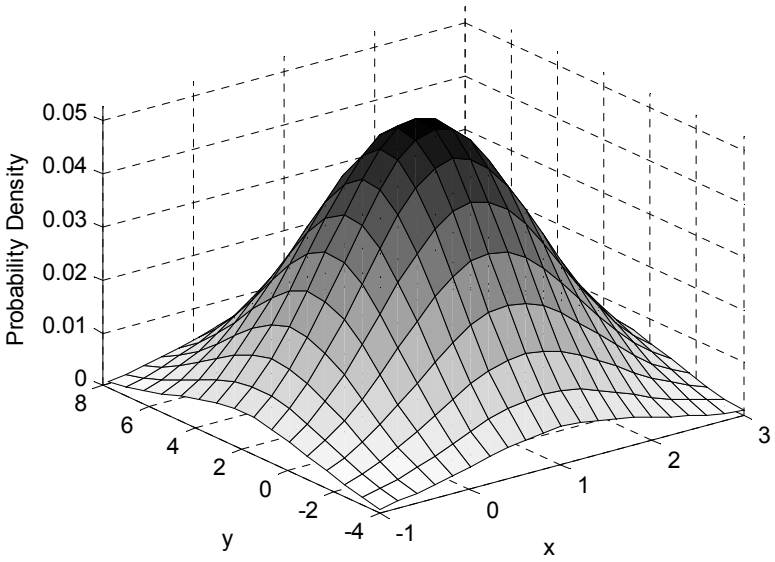


Figure 2.17 PDF of the bivariate distribution of the two independent normal variables.

Let $\underline{z}$ be a linear combination of $\underline{x}$ and $\underline{y}$, i.e. $\underline{z} = a\underline{x} + b\underline{y} + c$, where a, b and c are real numbers. Then, according to statistics and probability theory, $\underline{z}$ follows also a normal distribution with the following properties:

$$\mu_z = a\mu_x + b\mu_y + c \tag{2.62}$$

$$\sigma_z = \sqrt{(a\sigma_x)^2 + (b\sigma_y)^2} \tag{2.63}$$

For the numerical example's purposes, let $z=5x+4y+2$, thus according to Eqs. (2.62) and (2.63), $\mu_z=15$ and $\sigma_z=13$. Figure 2.18 depicts the probability density functions for the two independent variables x, y and the dependent variable z.

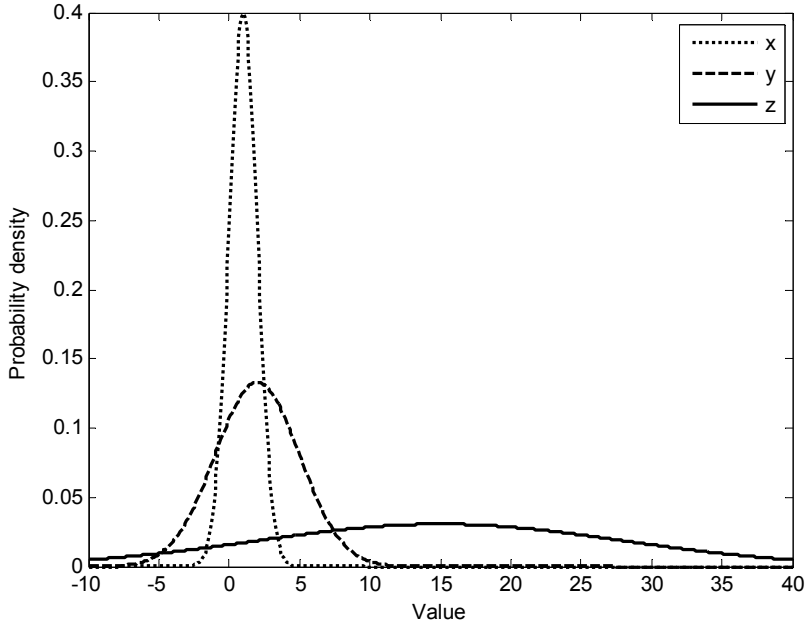


Figure 2.18 Probability density functions for the three variables *x*, *y*, *z*.

Assuming that the analytical relationships for the calculation of the mean value and the standard deviation of *z* (Eqs. (2.62) and (2.63)) are not known, we will try to calculate the statistical quantities of *z* numerically with crude MCS (MCS) and MCS with LHS (MCS-LHS) and the results will be compared to the exact analytical values.

Sample sizes starting from 100 samples and ending in 10000 samples with an increment of 100 are used. For a given sample size, *z*-random numbers are generated either purely randomly (Crude MCS) or by LHS (MCS-LHS). Every time, the mean value and the standard deviation are calculated. The figures below depict the calculated values versus the sample size for the mean and the standard deviation.

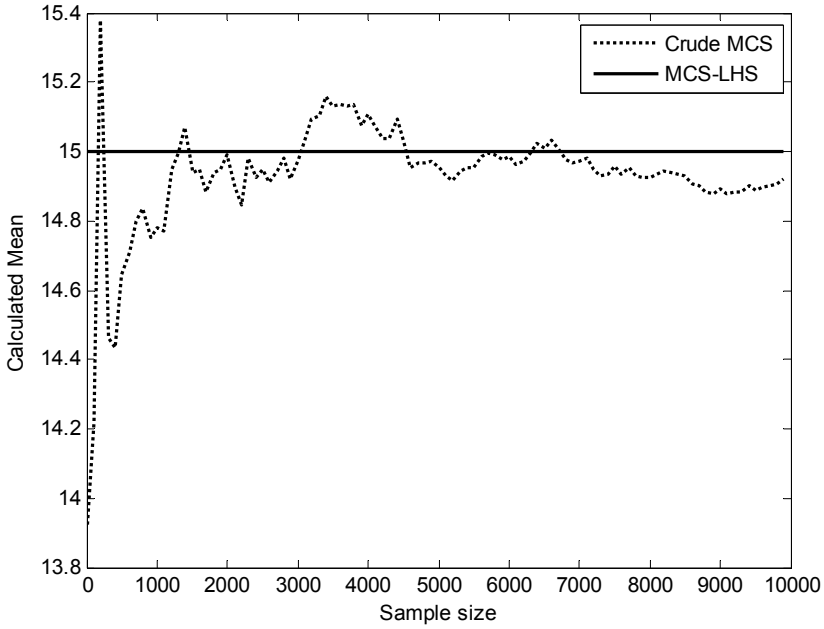


Figure 2.19 MC sampling for the normal distribution: Mean value vs Sample size.

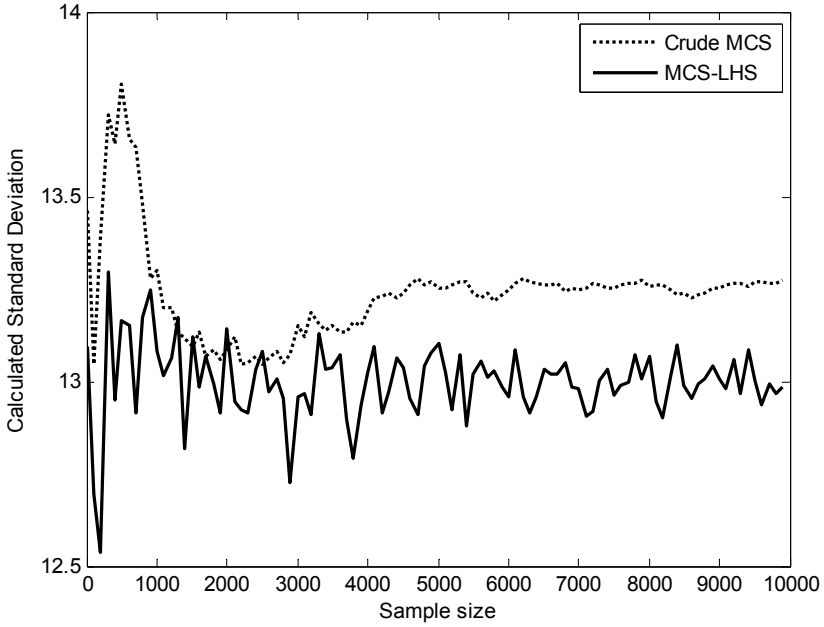


Figure 2.20 MC sampling for the normal distribution: Sigma vs Sample size.

The superiority of MCS-LHS over Crude MCS is clearly demonstrated also in the two-variable numerical test. For the calculation of the mean value, MCS needs too many iterations to converge to the exact value, while LHS needs only a few, as shown in Figure 2.19. For the calculation of the standard deviation, the performance of the MCS-LHS is

clearly better, as it is constantly closer to the correct value of 13 over all the range of sample sizes, as shown in Figure 2.20.

Note that the MCS line in Figure 2.20 is smoother than the MCS-LHS. This is due to the fact that in MCS, in order to increase the sample size, new random numbers can be generated and they can be added to the existing sample, so existing samples are always taken into account when the sample size is to be increased, as was discussed in detail in Section 2.8. The existing samples are taken into account for the calculation of the statistical values of larger samples and as a result a smoother line is obtained. On the contrary, in MCS-LHS, all samples have to be generated from the beginning and as a result the corresponding line is not smooth. Figure 2.21 shows the distribution of 1000 (x,y) samples in the x-y plane for the MCS and MCS-LHS cases.

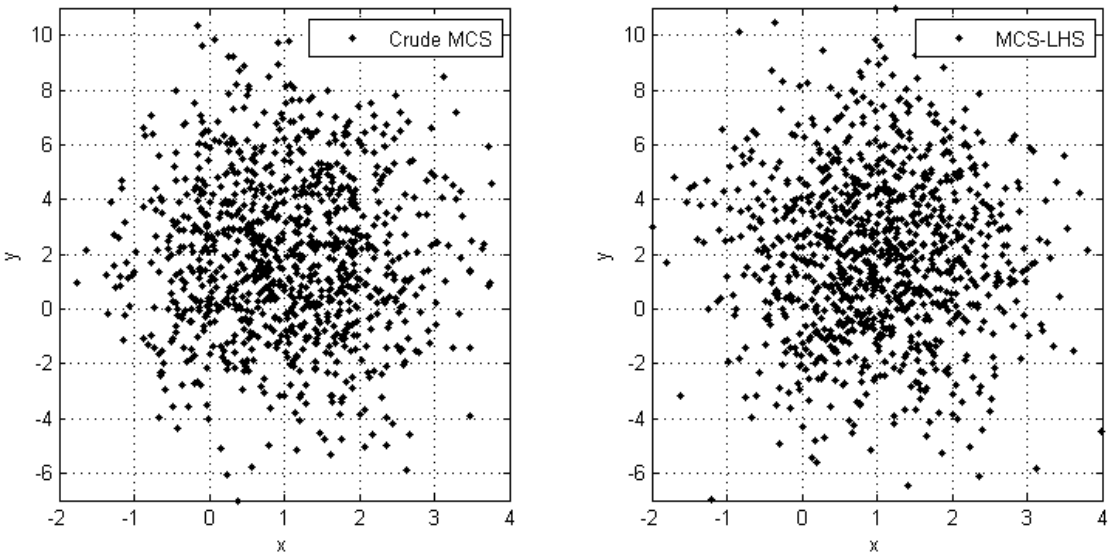


Figure 2.21 Distribution of 1000 samples in the x-y plane for (a) Crude MCS, (b) MCS-LHS.

2.9 Importance Sampling (IS)

The accurate estimation of probabilities of rare events through MCS is a very difficult task, as rare events are almost always defined on the tails of probability density functions. They have small probabilities and occur infrequently in real applications or during a simulation. This makes it difficult to generate them in sufficiently large numbers so that statistically significant conclusions can be drawn.

However, these events can be made to occur more often by deliberately introducing changes in the probability distributions that govern their behavior. Results obtained from such simulations are then altered to compensate for or undo the effects of these changes. Thus, in *Importance Sampling* (IS) (Anderson 1999; Srinivasan 2002), the main goal is to replace a sample distribution using another distribution that places more

weight in the areas of importance. Such a distribution function is problem-dependent and in most cases is difficult to find.

Importance Sampling is a general technique for estimating the properties of a particular distribution, while only having samples generated from a different distribution rather than the distribution of interest. The main idea behind the method is that certain values of the input random variables in a simulation have more impact on the parameter being estimated than others. If these "important" values are emphasized by sampling more frequently, then the estimator variance can be reduced. The basic methodology is to choose a distribution which "encourages" the important values. This leads to the use of a "biased" distribution which results in a biased estimator if it is applied directly in the simulation. However, the simulation outputs are weighted to correct for the use of the biased distribution, and this ensures that the new estimator is unbiased. The weight is given by the likelihood ratio (the ratio of the maximum probability of a result under two different hypotheses), that is, the Radon-Nikodym derivative (Shilov and Gurevich 1978) of the true underlying distribution with respect to the biased simulation distribution.

The fundamental issue in implementing importance sampling is the appropriate choice of the biased distribution which will encourage the important regions of the input variables. A good distribution can reduce significantly the computational time, in particular when estimating rare event probabilities (Rubinstein and Kroese 2008). On the other hand, a bad distribution can cause longer run times than the crude Monte Carlo Simulation technique without importance sampling. Let

$$\ell = \mathbb{E}_f\left(H(\underline{x})\right) = \int H(x)\cdot f(x)\mathrm{d}x \tag{2.64}$$

where H is the sample performance (e.g. the performance function) and f is the probability density function of the random variable $\underline{x}$. For clarification reasons, a subscript f is added to the expectation E (expected value) to indicate that it is taken with respect to the density function f. Let g be another probability density function such that the function $H{\cdot}f$ is dominated by function g, that is,

$$g(x) = 0 \quad \Rightarrow \quad h(x)\cdot f(x) = 0 \tag{2.65}$$

Using the density function g we can now represent ℓ as

$$\ell = \int \frac{H(x)\cdot f(x)}{g(x)} g(x)\mathrm{d}x = \mathbb{E}_g\left(\frac{H(\underline{x})\cdot f(\underline{x})}{g(\underline{x})}\right) \tag{2.66}$$

where the subscript g now means that the expectation is taken with respect to function g. Such a density is called the *importance sampling density*. Consequently, if $x_1, \ldots, x_n$ is a random sample from $\underline{g}$, then

$$\hat{\ell} = \frac{1}{n}\sum_{k=1}^{n} H(x_k)\cdot\frac{f(x_k)}{g(x_k)} \tag{2.67}$$

is an unbiased estimator of ℓ. This estimator is called the *importance sampling estimator*. The ratio of densities,

$$W(x) = \frac{f(x)}{g(x)} \tag{2.68}$$

is called the *likelihood ratio*. For this reason the importance sampling estimator is also called the *likelihood ratio estimator*. In the particular case where there is no change of measure, that is $g=f$, we have $W=1$, and the likelihood ratio estimator reduces to the usual MCS estimator.

It is important to realize that although Eq. (2.67) is an unbiased estimator for any PDF g dominating $H \cdot f$, not all such PDFs are appropriate. One of the main rules for choosing a good importance sampling PDF is that the estimator of Eq. (2.67) should have finite variance. This is equivalent to the requirement that

$$\mathbb{E}_g\left(H^2(\underline{x}) \cdot \frac{f^2(\underline{x})}{g^2(\underline{x})} \right) = \mathbb{E}_f\left(H^2(\underline{x}) \cdot \frac{f^2(\underline{x})}{g^2(\underline{x})} \right) < \infty \tag{2.69}$$

which suggests that g should not have a "lighter tail" than f and that, preferably, the likelihood ratio f/g should be bounded (Rubinstein and Kroese 2008).

The main drawback of the IS method is that it requires prior knowledge of the structural behavior in order to determine the most effective sampling region, which for many practical problems is not clearly identifiable.

2.10 Other sampling methodologies

2.10.1 Descriptive Sampling

In a Monte Carlo application, sampled distributions are assumed to be known. Using simple random sampling, sample histograms or, equivalently, sample moments will vary randomly, thus producing an imprecise description of the known input distribution, and consequently increasing the variance of simulation estimates. *Descriptive Sampling* (DS) (Saliby 1990; Saliby 1997) is a Monte Carlo Sampling technique based on a deterministic and purposive selection of the sample values and their random permutation in order to conform as closely as possible to the sampled distribution. Abandoning the 'principle' that sample values must be necessarily randomly generated in order to describe random behavior, DS is a rather polemical idea, based on a fully deterministic selection of the input sample values and their random permutation. DS avoids set variability of the input values and leads to more precise simulation estimates. However, since DS is based on a non-random selection of input sample values, it also represents a fundamental conceptual change on how to sample in any Monte Carlo application. The DS proposal questions the paradigm that a random selection of sample values would be more appropriate. It is justified by the fact that in any Monte Carlo application, the sampled distribution

must be assumed known a priori. As such, the sampling context is not inferential, where one seeks to acquire information about a population, but descriptive, where the purpose is just to describe the already known assumed PDF.

An interesting issue related to DS is its similarities to LHS, as DS can be seen as a limiting case of LHS. Like LHS, DS is based on a highly controlled selection of the input values and their random permutation. The difference between the two methods is that LHS still preserves a minimum random variability on the sample values selection, which is completely eliminated in DS.

2.10.2 Control Variates

In a Monte Carlo Simulation method, one or more control variates may be employed to achieve variance reduction by exploiting the correlation between statistics. This approach is called the *Control Variates* method (L'Ecuyer and Buist 2008; Szechtman 2003). Suppose $\underline{m}$ is a random variable with a mean value $\mathbb{E}(\underline{m})=\mu$ that is unknown, to be calculated with MCS. If one is able to find another statistic $\underline{t}$ such that the mean value $\mathbb{E}(\underline{t})=\tau$ is known, an unbiased estimator for μ is given by

$$\mu = \mathbb{E}(m^*) \tag{2.70}$$

where

$$m^* = m + c(t - \tau) \tag{2.71}$$

and c is a constant. It can be shown that in order to minimize the variance of m^*, the following value of the constant has to be used

$$c = -\frac{\operatorname{cov}(\underline{m}, \underline{t})}{\operatorname{var}(\underline{t})} = -\frac{\sigma_m}{\sigma_t}\rho_{mt} \tag{2.72}$$

where ρ_{mt} is the correlation between $\underline{m}$ and $\underline{t}$ and σ_m^2, σ_t^2 are the variances of $\underline{m}$ and $\underline{t}$, respectively. In this case, the variance of m^* is given by

$$\operatorname{var}(m^*) = (1 - \rho_{mt}^2) \cdot \operatorname{var}(m) \tag{2.73}$$

As shown in Eq. (2.73), the greater the value of the correlation $|\rho_{mt}|$, the greater the variance reduction achieved. In the case that σ_m, σ_t and ρ_{mt} are not known, they can be either somehow "guessed" or estimated with Monte Carlo Simulation.

2.10.3 Antithetic Variates

An alternative to using control variates is to consider the method of antithetic variates (Fishman and Huang 1983; Ross 2006). Suppose $\underline{m}$ is a random variable with a mean value $\mathbb{E}(\underline{m})=\mu$ that is unknown, to be calculated with MCS. Suppose $\underline{u}$ and $\underline{v}$ are two random variables with the same expected values $\mathbb{E}(\underline{u})=\mu$ and $\mathbb{E}(\underline{v})=\mu$. Then an unbiased estimator for μ is given by

$$\mu = \mathbb{E}\left(\frac{\underline{u} + \underline{v}}{2} \right) \tag{2.74}$$

with $$\operatorname{var}\left(\frac{\underline{u} + \underline{v}}{2} \right) = \frac{1}{4}\left(\operatorname{var}(\underline{u}) + \operatorname{var}(\underline{v}) + 2\operatorname{cov}(\underline{u}, \underline{v}) \right) \tag{2.75}$$

Thus, in order to calculate μ by sampling with the two random variables $\underline{u}$ and $\underline{v}$, it would be advantageous if $\underline{u}$ and $\underline{v}$ were negatively correlated, instead of being independent, as that would reduce the variance of the simulation, as shown in Eq. (2.75). Suppose that $\underline{u}$ can be written as a function of k random numbers uniformly distributed in the interval (0,1).

$$\underline{u} = h(\underline{w}_1, \ldots, \underline{w}_k) \tag{2.76}$$

If $\underline{w}_1, \ldots, \underline{w}_k$ follow the uniform distribution in the interval (0,1), then so are 1-$\underline{w}_1, \ldots, 1$-$\underline{w}_k$. Hence, the random variable

$$\underline{v} = h(1 - \underline{w}_1, \ldots, 1 - \underline{w}_k) \tag{2.77}$$

has the same distribution as $\underline{u}$. The random variables 1-$\underline{w}_1, \ldots, 1$-$\underline{w}_k$ are clearly negatively correlated with $\underline{w}_1, \ldots, \underline{w}_k$. It can be proved that, in the special case where h is a monotone (either increasing or decreasing) function of each of its coordinates, also $\underline{u}$ is negatively correlated with $\underline{v}$. Thus, in this special case, after $\underline{w}_1, \ldots, \underline{w}_k$ are generated as to compute $\underline{u}$, rather than generating a new independent set of random variables, one can do better by just using the set 1-$\underline{w}_1, \ldots, 1$-$\underline{w}_k$ to compute $\underline{v}$. This is the main concept of the antithetic variates technique.

By using this technique, the benefit is double, as: (i) the resulting estimator has reduced variance, provided that h is a monotone function; and (ii) there is no need to generate a second set of random numbers, which also saves computational time.

2.10.4 Adaptive Sampling

Adaptive Sampling (Thompson and Seber 1996), is a sampling scheme in which sampling regions are selected based on values of the variables of interest observed during a sampling survey. In a conventional sampling scheme, the selection for a sampling point does not depend on previous observations made during an initial survey; the entire sampling set is selected, before any physical sampling in the field ever takes place. Therefore, conventional sampling guarantees that the calculated statistics will be unbiased.

On the other hand, adaptive procedures allow adjustments as the sampling is being carried out. This way, Adaptive Sampling can make more efficient use of resources without diminishing the statistical power of the procedure. If additional units are added to the sampling design wherever high positive identifications are observed, the sample mean will over-estimate the population mean. This way, Adaptive Sampling introduces biases into conventional estimators, so new unbiased estimators are needed (Thompson 2002).

A method of obtaining unbiased estimators is to make use of new observations in addition to the observations initially selected.

2.10.5 Hammersley Sequence Sampling (HSS)

Latin hypercubes are designed for uniformity along a single dimension where subsequent columns are randomly paired for placement on an *m*-dimensional cube. Therefore, the likelihood of such schemes providing good uniformity properties on high-dimensional cubes is small. *Hammersley Sequence Sampling* (HSS), based on Hammersley points, provides a low-discrepancy experimental design for placing *N* points in an *m*-dimensional hypercube (Kalagnanam and Diwekar 1997; Wang et al. 2004), ensuring that the sample set is more representative of the population and providing better uniformity properties over the *m*-dimensional space, unlike other sampling techniques. A low discrepancy implies a uniform distribution of points in space.

One of the main advantages of MCS is that the number of samples required to obtain a given accuracy of estimates does not scale exponentially with the number of uncertain variables, a characteristic that is preserved also in HSS. For correlated samples, the approach described by Kalagnanam and Diwekar (1997) for HSS uses rank correlations to preserve the stratified design along each dimension. By imposing the correlation structures, the uniformity property of the scheme is preserved, but the optimal location of the Hammersley points is perturbed. HSS is generated based on prime numbers as bases. Although the original HSS technique designs start at the same initial point, they can be randomized by choosing the first prime number randomly. It has been found that the uniformity property of HSS for higher dimensions (more than 30 uncertain variables) gets distorted. In order to break this distortion, leaps in prime numbers can be introduced for higher dimensions. The leaped HSS circumvents the distortion at higher dimensions. According to Kalagnanam and Diwekar (1997) the HSS technique is considered more efficient than crude MCS and LHS for uncertainty analysis.

Chapter 3

3 Single-objective Optimization

3.1 The concept of optimum structural design

Generally, a structure has to serve a specific purpose, under some constraints. Engineering structures such as bridges, aircrafts or buildings have to be built in an optimum way, in order to be safe and also economic. The optimum design is the main goal of an engineer who aims at designing a machine part or a structure. Until a few decades ago, the fulfillment of this purpose was very difficult since no automated algorithms or computer methods were available. Traditionally, the design of a structure was done via some kind of trial and error. Since then, many methods have been proposed in the literature to solve structural optimization problems. Each method exhibits some advantages and some disadvantages.

The rapid development of computer technology nowadays has led to an increase in the demands from our structures. A structure that merely fulfils the safety or other requirements is not adequate. The aim is to construct the system optimally, in order to fulfill the safety requirements on one hand and on the other hand to minimize some criterion such as the cost or the weight of the structure. The design of complicated structural systems requires a number of time consuming calculations. The development and implementation of appropriate algorithms has made possible the optimum design of large-scale structural systems. By utilizing the available technology, now the engineer can design better structures, with reduced construction and operation cost of in an affordable computing time.

Two steps are required in order to solve an optimization problem. The first is the mathematical formulation of the optimization problem and the second is the computer application of an optimization algorithm. The first step has to do with the definition of the design variables, the definition of an objective function and the corresponding constraint functions. The optimization finishes with the selection of an appropriate optimization algorithm for the specific problem, in combination with a finite element model for the structure.

The application of an advanced optimization algorithm on a specific engineering problem is not enough by itself. The experience of the engineer plays important role in order to use the algorithms efficiently. The role of the engineer is to formulate the problem in an appropriate way, in order for the automated optimization process to give valuable results.

3.2 Types of structural optimization problems

There are mainly three classes of structural optimization problems: (i) sizing; (ii) shape; and (iii) topology optimization. Figure 3.1 gives a graphical representation of the three classes of problems.

3.2.1 Sizing Optimization

In sizing optimization problems the aim is mainly to minimize the weight of the structure under certain behavioral constraints on stresses and displacements. The design variables are most frequently chosen to be dimensions of the cross-sectional areas of the members of the structure. Due to engineering practice demands the members are divided into groups having the same design variables. This grouping of elements results in a trade-off between the use of more material and the need of symmetry and uniformity of structures due to practical considerations. Furthermore, it has to be taken into account that due to fabrication limitations the design variables may not be continuous but discrete since cross-sections belong to a certain predefined set, provided by the manufacturers.

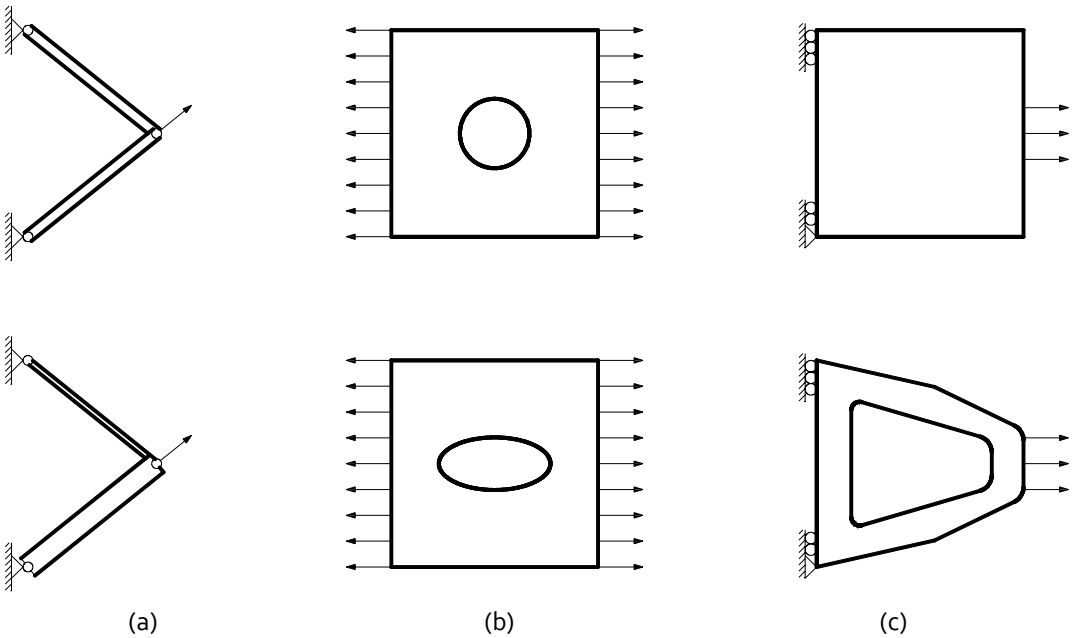


Figure 3.1 Classes of structural optimization problems:
(a) Sizing, (b) Shape and (c) Topology optimization.

3.2.2 Shape Optimization

In structural shape optimization problems the aim is to improve the performance of the structure by modifying its boundaries and therefore its shape. This can be numerically achieved by minimizing an objective function subjected to certain constraints (Hinton and Sienz 1994; Ramm et al. 1994). The design variables are either some of the coordinates of the key points in the boundary of the structure or some other parameters that influence the shape of the structure. When shape optimization is considered, the structural domain is not fixed but has a predefined topology.

3.2.3 Topology Optimization

Structural topology optimization assists the designer to define the type of structure, which is best suited to satisfy the operating conditions for the problem at hand. It can be seen as a procedure of optimizing the rational arrangement of the available material in the design space and eliminating the material that is not needed. Topology optimization is usually employed in order to achieve an acceptable initial layout of the structure, which is then refined with a shape optimization tool. Various methods have been proposed for topology optimization problems, employing the following main approaches (Hinton and Sienz 1993): (i) Ground structure approach (Pedersen 1993; Schwefel 1981); (ii) homogenization method (Bendsoe and Kikuchi 1988; Hinton and Hassani 1995; Suzuki and Kikuchi 1993); (iii) bubble method (Eschenauer et al. 1993); and (iv) fully stressed design technique (Van Keulen and Hinton 1996; Xie and Steven 1993). The first three approaches behave as normal optimization techniques. On the other hand, the fully stressed design technique is not an optimization algorithm in the conventional sense, as it proceeds by removing inefficient material, and therefore optimizes the use of the remaining material in the structure, in an evolutionary process.

3.3 Formulation of a single-objective optimization problem

The formulation of the generic *Single-objective Optimization Problem* (SOP) can be written as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}), \quad \boldsymbol{x} = [x_1,\ldots,x_n]^{\mathrm{T}}, \quad f \in \mathbb{R} \\ &\text{Subject to} \\ &\boldsymbol{g}(\boldsymbol{x}) \le 0, \quad \boldsymbol{g} \in \mathbb{R}^p \\ &\boldsymbol{h}(\boldsymbol{x}) = 0, \quad \boldsymbol{h} \in \mathbb{R}^q \\ &x_i \in X_i \quad \text{for } i = 1,\ldots,n \end{aligned} \tag{3.1}$$

where:

- $\boldsymbol{x} = [x_1,\ldots,x_n]^{\mathrm{T}}$ is a vector of length n containing the design variables.

- X_i is the set of x_i, which may be continuous, discrete or integer. The whole design space for the n design variables can be denoted as $\mathcal{X}$.
- $f(\boldsymbol{x})$: $\mathbb{R}^n \to \mathbb{R}$ is the objective function, which returns a scalar value to be minimized.
- $\boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} = [g_1(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x})]$ is the vector function of p inequality constraints.
- $\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}} = [h_1(\boldsymbol{x}), \ldots, h_q(\boldsymbol{x})]$ is the vector function of q equality constraints.

In structural design optimization, inequality constraints are mainly used, since equality constraints are not applicable for real-world problems. If the objective function is the weight of the structure, then it is given by

$$f(\boldsymbol{x}) = \rho \cdot \sum_{i=1}^{N_e} A_i \cdot L_i \tag{2}$$

where ρ is the material density, N_e is the number of elements of the model and A_i, L_i are the cross sectional area and the length of each element, respectively.

3.3.1 Discrete and continuous formulations

In structural design optimization, due to manufacturing limitations, the design variables are not described by continuous functions but are discrete variables (Makris et al. 2006) since cross-sections have to belong to a certain predefined set provided by the manufacturers. There are also cases where for the same problem the design variables are mixed, continuous and discrete, e.g. in a topology-sizing optimization problem where the design variables include nodal coordinates (continuous) as well as beam cross-sectional sizes (discrete).

With the general formulation of Eq. (3.1), the design variables may have continuous, discrete or integer values, or a combination of them, with the restriction

$$x_i \in X_i \quad \text{for } i = 1, \ldots, n \tag{3.3}$$

where X_i is the set of x_i, which may be continuous or discrete. When discrete design variables are only used, then the available set of values is clearly defined. When continuous design variables are considered, then the above restriction is usually written as

$$\boldsymbol{x}^{\mathrm{L}} \le \boldsymbol{x} \le \boldsymbol{x}^{\mathrm{U}} \tag{3.4}$$

where $\boldsymbol{x}^{\mathrm{L}}$ and $\boldsymbol{x}^{\mathrm{U}}$ are two vectors of length n containing the lower and upper bounds of the design variables, respectively.

Various methods have been proposed for dealing with mixed problems, with continuous and discrete design variables (Bremicker et al. 1990). Usually discrete variables are han-

dled as equivalent continuous variables, and at the end of the optimization process the design variables are given the appropriate discrete values, as close as possible to the optimal continuous values (Hager and Balling 1988). In case of a discrete problem where the design space can be univocally arranged for all the characteristics of the cross sections, the above method can give a good approximation of the discrete optimum solution. Nevertheless, in realistic engineering problems this may not be the case. Most of the methods that have been proposed convert the mixed problem to a series of continuous problems that are solved consecutively (Cai and Thierauf 1993a; Cai and Thierauf 1993b; Fu et al. 1991).

3.4 Definitions

Convex set: Let $\mathcal{C}$ be a set in a real or complex vector space. $\mathcal{C}$ is said to be a *convex set* if, for all $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathcal{C}$ and all t in the interval [0,1], the point

$$(1-t)\cdot\boldsymbol{x}+t\cdot\boldsymbol{y} \tag{3.5}$$

is also in $\mathcal{C}$. In other words, every point on the line segment connecting $\boldsymbol{x}$ and $\boldsymbol{y}$ is in $\mathcal{C}$.

Convex function: A real-valued function f defined on an interval is called *convex* (or *concave upwards*, *concave up* or *convex cup*), if for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in its domain and any t in the interval [0,1],

$$f(t\cdot\boldsymbol{x}+(1-t)\cdot\boldsymbol{y})\le t\cdot f(\boldsymbol{x})+(1-t)\cdot f(\boldsymbol{y}) \tag{3.6}$$

In other words, a function is convex if and only if its epigraph (the set of points lying on or above the graph) is a convex set.

Strictly convex function: A real-valued function f defined on an interval is called *strictly convex*, if for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in its domain with $\boldsymbol{x}\neq\boldsymbol{y}$ and any t in the interval (0,1),

$$f(t\cdot\boldsymbol{x}+(1-t)\cdot\boldsymbol{y})< t\cdot f(\boldsymbol{x})+(1-t)\cdot f(\boldsymbol{y}) \tag{3.7}$$

Pictorially, a function is called convex if the function lies below the straight line segment connecting two points, for any two points in the interval. A convex function of one variable is depicted in Figure 3.2(a), namely the function $f(x)=(x\text{-}5)^2+5$ in the interval $x\in[0,10]$. It is clear that the function lies below the straight line segment connecting points A and B, for any two points in the interval.

A non-convex function of one variable is depicted in Figure 3.2(b), namely the function $f(x)=(x\text{-}1)\cdot(x\text{-}4)\cdot(x\text{-}5)\cdot(x\text{-}7)$ in the interval $x\in[0,8]$. It is clear that there are points in the interval for which the function does not lie below the straight line segment connecting these points, e.g. points *A* and *B*. At first, the function lies above the segment, while after a specific point the function lies below the segment.

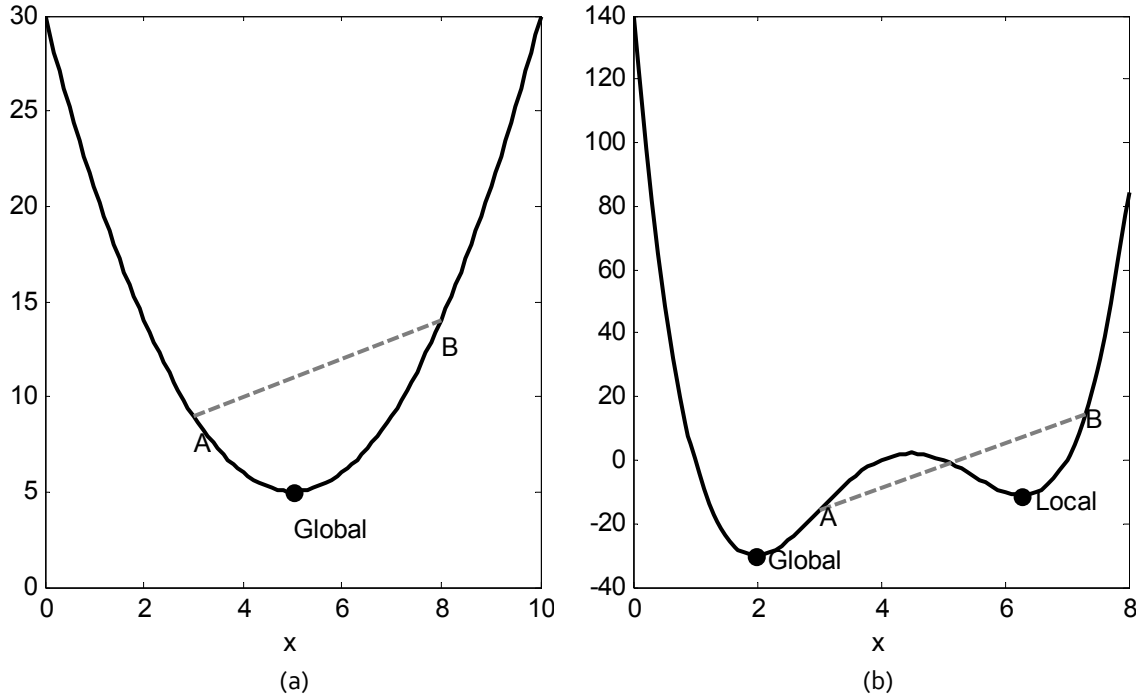


Figure 3.2 (a) A convex one-variable function with a global minimum, (b) A non-convex one-variable function with a global and a local minimum.

Concave function: A real-valued function f defined on an interval is called *concave* (or *concave downwards*, *concave down* or *convex cap*) if $-f$ is convex.

Feasible set: The *feasible set* $\mathcal{F}$ is defined as the set of design vectors $\boldsymbol{x}$ that satisfy the constraints of the optimization problem of Eq. (3.1). It is obvious that $\mathcal{F}$ is a subset of $\mathcal{X}$.

$$\mathcal{F} = \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{g}(\boldsymbol{x}) \leq 0 \mid \boldsymbol{h}(\boldsymbol{x}) = 0\} \tag{3.8}$$

$$\mathcal{F} \subseteq \mathcal{X} \tag{3.9}$$

The image of $\mathcal{F}$, i.e. the feasible region in the objective space, is called the *criterion space*, denoted as $\mathcal{Y}_{\mathcal{F}}=f(\mathcal{F})$. All equality constraints (regardless of the value of $\boldsymbol{x}$ used) are considered active at all points of the feasible set $\mathcal{F}$.

Feasible design: A design vector $\boldsymbol{x}$ is feasible if and only if it belongs to the feasible set $\mathcal{F}$.

Global minimizer: A design vector $\boldsymbol{x}^* \in \mathcal{F}$ is said to be a global minimizer of the minimization problem of Eq. (3.1) if and only if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \ \forall \ \boldsymbol{x} \in \mathcal{F}$. The corresponding value of the objective function $f(\boldsymbol{x}^*)$ is called a *global minimum*.

Local minimizer: A design vector $\boldsymbol{x}^{\mathrm{L}} \in \mathcal{F}$ is said to be a local minimizer of the optimization problem of Eq. (3.1) if there exists a neighborhood $\aleph$ of $\boldsymbol{x}^{\mathrm{L}}$ such that

$f(\boldsymbol{x}^{\mathrm{L}}) \leq f(\boldsymbol{x}) \ \forall \ \boldsymbol{x} \in \aleph$. The corresponding value of the objective function $f(\boldsymbol{x}^{\mathrm{L}})$ is called a *local minimum*.

From the above definitions it is clear that a global minimizer is necessarily also a local minimizer, while the converse does not hold in general. The converse is true only in the special case where all the functions involved are convex functions resulting in a unique local minimum that is also the global minimum.

The convex function of one variable $f(x)=(x\text{-}5)^2+5$ shown in Figure 3.2(a) has a local minimizer with $f_{\min}=5$ at point x=0, which is also a global minimizer. On the contrary, the non-convex function of one variable $f(x)=(x\text{-}1)\cdot(x\text{-}4)\cdot(x\text{-}5)\cdot(x\text{-}7)$ shown in Figure 3.2(b) has two local minimizers at points $x_1\approx1.9765$ and $x_2\approx6.2873$, and a global minimizer at point $x_1\approx1.9765$, which is also a local minimizer.

A convex function of two variables is depicted in Figure 3.3, namely the function $f(x,y)=x^2+y^2$ in the region $x\in[-4,4]$, $y\in[-4,4]$, with a global minimizer at point $x=0$, $y=0$, resulting in $f_{\min}=0$.

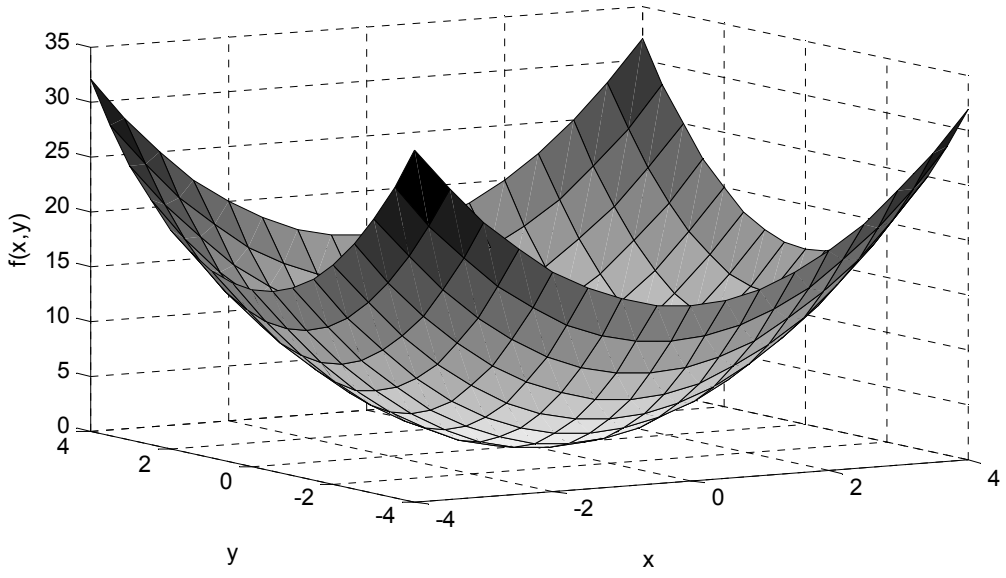


Figure 3.3 A convex function of two variables with a single optimum.

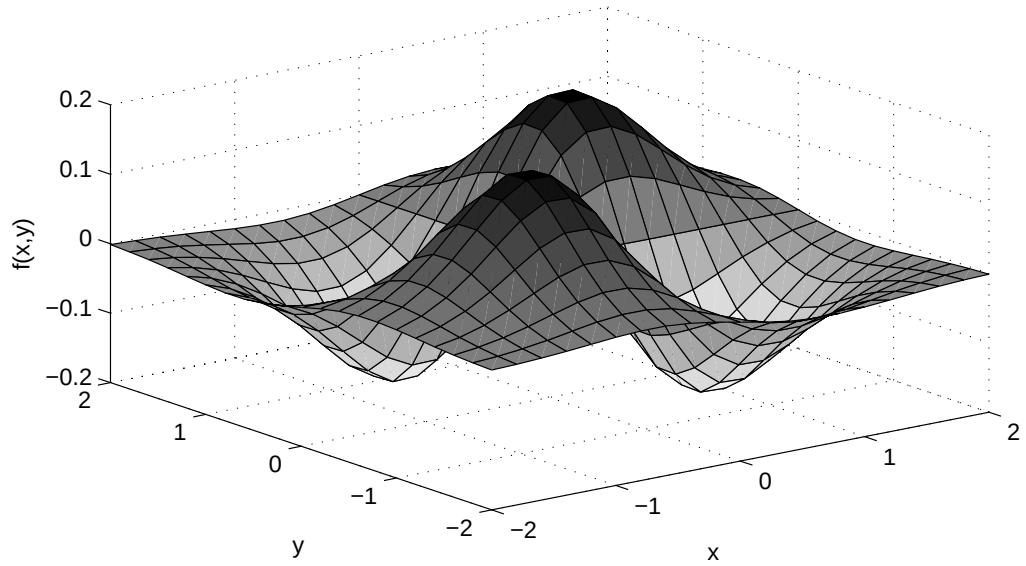


Figure 3.4 A non-convex function of two variables with multiple local optima.

A non-convex function of two variables is depicted in Figure 3.4, namely the function $f(x,y)=x\cdot y\cdot\exp(-x^2-y^2)$ in the region $x\in[-2,2]$, $y\in[-2,2]$. As shown in the figure, this function has multiple local minimizers.

3.5 Methods for solving SOPs

Several numerical methods have been developed during the last three decades to meet the demands of structural design optimization. These methods can be widely classified into two general categories:

i. Deterministic (or mathematical) optimization methods;

ii. Probabilistic optimization methods.

Detailed descriptions of the two families of methods will be given in the following sections. According to the *No Free Lunch* (NFL) theorems for optimization (Wolpert and Macready 1997), for any algorithm, any elevated performance over one class of problems is offset by performance over another class. These theorems result in a geometric interpretation of what it means for an algorithm to be well suited to an optimization problem. In other words, there is no optimization algorithm capable of handling efficiently every optimization problem.

In the present thesis, probabilistic optimization algorithms are mainly used and in particular Evolution Strategies and *Particle Swarm Optimization* (PSO), while a hybrid algorithm combining PSO and SQP is also proposed for structural optimization, combining the advantages of probabilistic and mathematical methods.

3.6 Mathematical Programming

The first family of optimization algorithms used in structural engineering were taken from other scientific fields such as economics, mathematics and operational research and belong to the *Mathematical Programming* (MP) methods. MP methods, and in particular the gradient-based optimizers that have been basically applied for solving structural optimization problems in the past, belong to the deterministic optimization methods (Lamberti and Pappalettere 2004; Thanedar et al. 1986).

Mathematical programming (gradient-based) optimization methods are generally considered as local methods. Algorithms belonging to this category, such as the *Sequential Quadratic Programming* (SQP) method (Belegundu and Arora 1985a; Belegundu and Arora 1985b; Papadrakakis et al. 1996b; Thanedar et al. 1986), the *Generalized Reduced Gradient* (GRG) method (Lasdon et al. 1978), the *Method of Moving Asymptotes* (MMA) (Svanberg 1987) and the *Method of Feasible Directions* (MFD) (Svanberg 1987) have been used in the past for structural optimization problems. They make use of local curvature information and require both objective function and gradient evaluations.

The main advantage of deterministic methods is that, by using gradient information in order to monitor and guide the search, the convergence rate is faster towards the optimum. In the case of the SQP method, the convergence rate is exponential, due to the second order information that is inherent in the gradient information. Sometimes however, even small violations of the non-linear constraint functions can delay the convergence, while there is no guarantee that the method will converge to the global optimum without being trapped in a local optima. Although many mathematical optimization applications have no constraints, optimization problems in the engineering field usually have some kind of constraints, as the result of the space within which the design variables have to stay, or other constraint functions, having to do with stresses, displacements, etc.

Mathematical programming algorithms are still being developed and improved. Lately, a new class of algorithms, namely the *Trust-Region Method* (TRM) (Conn et al. 2000) has been developed for dealing with unconstrained or constrained optimization problems. A trust region method approximates only a certain region (the so-called *trust region*) of the objective function with a quadratic function, as opposed to the entire function as with the Newton-Raphson optimization algorithm. The size of the region is modified during the search, based on how well the model agrees with actual function evaluations. When an adequate model of the objective function is found within the trust region, then the region is expanded. Conversely, if the approximation is poor then the region is contracted. The evaluation method is to observe the ratio of expected improvement from the quadratic approximation with the actual improvement observed in the objective function. Trust region methods can be considered as dual to line search methods, as they first choose a step size (the size of the trust region) and then a step direction while line search methods do the inverse.

The main advantage of MP optimizers is that they capture very fast the right path to the nearest optimum by exploiting gradient information, but they cannot guarantee the estimation of the global optimum, as they can be easily trapped in local optima. These methods most of the times require user-defined initial estimates of the solution in order to give results of good quality.

Usually, a real-world structural optimization problem, whether it is topology, shape, sizing or an integrated structural optimization problem, is a computationally intensive task, where most of the computations are spent for the solution of the finite element equilibrium equations required for the analysis steps within the optimization procedure. Usually real-world structural optimization applications do not favor the use of MP algorithms, because gradient-based optimizers typically encounter great difficulties in dealing with multiple local optima, large and non-convex search spaces and several (possibly conflicting) constraints to be satisfied (Nocedal and Wright 1999).

3.6.1 Sequential Quadratic Programming (SQP)

The mathematical optimizer adopted in this thesis is a SQP method. SQP methods are regarded as the standard general purpose mathematical programming algorithms for solving *Non-Linear Programming* (NLP) optimization problems (Gill et al. 1981). They are also considered to be the most suitable methods for solving structural optimization problems with the mathematical programming approach (Arora 1990; Provatidis and Venetsanos 2006; Schittkowski et al. 1994; Thanedar et al. 1986). Such methods make use of local curvature information derived from linearization of the original functions, by using their derivatives with respect to the design variables at points obtained in the process of optimization.

Given the problem description of Eq. (3.1) and considering a continuous problem with the restrictions of Eq. (3.4), SQP method proceeds with the conversion of the NLP problem into a sequence of *Quadratic Programming* (QP) subproblems based on a quadratic approximation of the Lagrangian function.

$$L(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{k=1}^{m} \lambda_k g_k(\boldsymbol{x}) \tag{3.10}$$

where λ_k are the Lagrange multipliers under the non-negativity restriction for the inequality constraints. The QP subproblem can be obtained by linearizing the non-linear constraints. Each QP subproblem has the following form:

$$\begin{aligned} &\min_{\boldsymbol{p}\in\mathbb{R}^n} \left\{ \frac{1}{2}\boldsymbol{p}^{\mathrm{T}}\boldsymbol{H}_\ell \boldsymbol{p} + \nabla f(\boldsymbol{x}_\ell)^{\mathrm{T}}\boldsymbol{p} \right\} \\ &\text{Subject to} \\ &\nabla g_k(\boldsymbol{x}_\ell)^{\mathrm{T}}\boldsymbol{p} + g_k(\boldsymbol{x}_\ell) \le 0 \quad k = 1,\ldots,m \end{aligned} \tag{3.11}$$

where $\boldsymbol{p}$ is the search direction and $\boldsymbol{H}_\ell$ a positive definite approximation of the Hessian matrix of the Lagrangian function of Eq. (3.10). The formulation of Eq. (3.11) considers only the inequality constraints, as equality constraints are not usually encountered in structural optimization problems. In order to construct the Jacobian and the Hessian matrices of the QP subproblem, the derivatives of the objective and constraint functions are required. These derivatives can be calculated during the sensitivity analysis phase either analytically, using a closed form if available, semi-analytically or numerically with a global finite difference method. In the present thesis, the latter option is implemented.

An estimate of the Hessian of the Lagrangian function is updated at each iteration using the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) (Broyden 1970; Fletcher 1970; Goldfarb 1970; Shanno 1970) quasi-Newton formula and a line search is performed using a merit function in order to determine the step length parameter. The QP subproblem is solved using an active set strategy. Constrained quasi-Newton methods guarantee superlinear convergence by accumulating second-order information regarding the *Karush-Kuhn-Tucker* (KKT) equations using a quasi-Newton updating procedure. The KKT equations are necessary conditions for optimality for a constrained optimization problem.

There are two ways to solve the QP subproblem, either with a primal (Gill et al. 1986), or a dual (Fleury 1993) formulation. The primal algorithm adopted in this study is divided into three phases:

i. Solution of the QP subproblem to obtain the search direction;

ii. Line search along the search direction;

iii. Update of the Hessian matrix.

Once the direction vector $\boldsymbol{p}$ is found a line search is performed in order to produce a sufficient decrease to the merit function Φ, discussed in the next paragraphs.

Line search and Merit function

At each step of the SQP algorithm, the solution of the QP subproblem of Eq. (3.11) produces the search direction $\boldsymbol{p}_\ell$. Then, the line-search method searches along the line containing the current point, $\boldsymbol{x}_\ell$, parallel to the search direction and the new design point is calculated as

$$\boldsymbol{x}_{\ell+1} = \boldsymbol{x}_\ell + a_\ell \cdot \boldsymbol{p}_\ell \tag{3.12}$$

The step length parameter a_ℓ is determined by the line search procedure so that sufficient decrease in a merit function is obtained. A merit function of the form

$$\Phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{k=1}^{m} r_k \cdot \max\{0, g_k(\boldsymbol{x})\} \tag{3.13}$$

is used, where r_k are the penalty parameters

$$r_k = (r_{\ell+1})_k = \max_k \left\{ \lambda_k, \frac{(r_\ell)_k + \lambda_k}{2} \right\}, \quad k = 1, \ldots, m \tag{3.14}$$

The penalty parameter r_k is initially set to

$$r_k = \frac{\|\nabla f(\boldsymbol{x})\|}{\|g_k(\boldsymbol{x})\|} \tag{3.15}$$

Update of the Hessian Matrix

At each major iteration a positive definite quasi-Newton approximation of the Hessian matrix of the Lagrangian function H is calculated using the BFGS method (Gill et al. 1981). Attention is given to keep the Hessian matrix positive definite. The Hessian matrix is updated using the following formula

$$\boldsymbol{H}_{\ell+1} = \boldsymbol{H}_\ell + \frac{\boldsymbol{q}_\ell \cdot \boldsymbol{q}_\ell^{\mathrm{T}}}{\boldsymbol{q}_\ell^{\mathrm{T}} \cdot \boldsymbol{s}_\ell} - \frac{\boldsymbol{H}_\ell^{\mathrm{T}} \cdot \boldsymbol{s}_\ell^{\mathrm{T}} \cdot \boldsymbol{s}_\ell \cdot \boldsymbol{H}_\ell}{\boldsymbol{s}_\ell^{\mathrm{T}} \cdot \boldsymbol{H}_\ell \cdot s_\ell} \tag{3.16}$$

where ℓ denotes the current SQP iteration and

$$\boldsymbol{s}_\ell = \boldsymbol{x}_{\ell+1} - \boldsymbol{x}_\ell \tag{3.17}$$

$$\boldsymbol{q}_\ell = \left(\nabla f(\boldsymbol{x}_{\ell+1}) + \sum_{k=1}^{m} \lambda_k \cdot \nabla g_k(\boldsymbol{x}_{\ell+1}) \right) - \left(\nabla f(\boldsymbol{x}_\ell) + \sum_{k=1}^{m} \lambda_k \cdot \nabla g_k(\boldsymbol{x}_\ell) \right) \tag{3.18}$$

The Hessian is maintained provided $\boldsymbol{q}_\ell^{\mathrm{T}} \cdot \boldsymbol{s}_\ell$ is positive at each update and that $\boldsymbol{H}$ is initialized as positive definite. If the quadratic function is convex then the Hessian is positive definite, or positive semi-definite and the solution obtained will be a global optimum. Else, if the quadratic function is non-convex then the Hessian is indefinite and if a solution exists it is only a local optimum.

3.6.2 Sensitivity Analysis

The most time-consuming part of an optimization algorithm based on mathematical programming methods is devoted to the sensitivity analysis phase (Papadrakakis et al. 1996b), which is an important part of all mathematical programming optimization methods. Although sensitivity analysis is mostly mentioned in the context of structural optimization, it has evolved into a research topic of its own. The calculation of the sensitivity coefficients follows the application of a relatively small perturbation to each primary design variable. Several techniques have been developed which can be mainly distinguished by their numerical efficiency and their implementation aspects (Bletzinger et al. 1991). They can be divided into two main categories: (i) discrete; and (ii) variational.

According to the discrete methods, the sensitivities, i.e. the gradients of the characteristic functions (displacements, stresses, etc) with regard to the design variables are calculated based on the equilibrium equations that derive from the discretization of the mod-

el with finite elements. A classification of the discrete methods for sensitivity analysis is the following:

i. ***Global Finite Difference* (GFD) method**: The method is simple in its formulation and can be programmed easily, yet it is the most time-consuming, as a full finite element analysis has to be performed for each design variable. The accuracy of the method depends strongly on the value of the perturbation of the design variables. The method exhibits the advantage that it can be easily applied to any kind of problem, even for finite element models with various types of elements. Also, it can be used with commercial finite element programs, as there is no need for modifications within the source code of the program.

ii. ***Semi-Analytical* (SA) method**: The stiffness matrix of the initial finite element solution is retained during the computation of the sensitivities. This provides an improved efficiency over the GFD method by a relatively small increase in the algorithmic complexity. The accuracy problem involved with the numerical differentiation can be overcome by using the "exact" semi-analytical method which needs more programming effort than the simple method but it is computationally more efficient.

iii. **Analytical method**: The finite element equations, the objective and constraint functions are differentiated analytically.

The finite difference and the semi-analytical approaches are the two most widely used types of sensitivity analysis techniques for structural optimization. From the algorithmic point of view, the semi-analytical technique results in a typical linear solution problem with multiple right-hand sides in which the stiffness matrix remains unchanged, while the finite difference technique results in a typical reanalysis problem in which the stiffness matrix is modified due to the perturbations of the design variables.

In structural optimization problems 60% to 90% of the computations are spent for the solution of equilibrium equations required for the finite element analysis and sensitivity analysis (Papadrakakis and Tsompanakis 1999; Papadrakakis et al. 1996b). Therefore, it is important to use efficient algorithms for both the solution of the equilibrium equations and the sensitivity analysis.

The Semi-Analytical Method

The semi-analytical method is based on the chain rule differentiation (Vanderplaats 1984) of the finite element equations

$$\boldsymbol{K} \cdot \boldsymbol{u} = \boldsymbol{f} \quad \Rightarrow \tag{3.19}$$

$$\boldsymbol{K} \frac{\partial \boldsymbol{u}}{\partial x_k} + \frac{\partial \boldsymbol{K}}{\partial x_k} \boldsymbol{u} = \frac{\partial \boldsymbol{f}}{\partial x_k} \tag{3.20}$$

which when rearranged results in

$$\boldsymbol{K}\frac{\partial \boldsymbol{u}}{\partial x_k} = \boldsymbol{f}_k^* \tag{3.21}$$

where $\boldsymbol{f}_k^*$ represents a pseudo-load vector

$$\boldsymbol{f}_k^* = \frac{\partial \boldsymbol{f}}{\partial x_k} - \frac{\partial \boldsymbol{K}}{\partial x_k}\boldsymbol{u} \tag{3.22}$$

The derivative of the stiffness matrix $\partial \boldsymbol{K}/\partial x_k$ and the derivative of the load vector $\partial \boldsymbol{f}/\partial x_k$ are computed approximately for each design variable by recalculating the new values of $\boldsymbol{K}(x_k + \Delta x_k)$ and $\boldsymbol{f}(x_k + \Delta x_k)$ for a small perturbation Δx_k of the design variable x_k. The derivatives of $\partial \boldsymbol{f}/\partial x_k$ are computed using a forward finite difference scheme. With respect to the differentiation of $\boldsymbol{K}$, the semi-analytical approach can be implemented in two versions: the *Conventional Semi-Analytical* (CSA) and the *Exact Semi-Analytical* (ESA) method.

In the Conventional Semi-Analytical method, the values of the derivatives in Eq. (3.22) are calculated by applying the forward difference approximation scheme

$$\frac{\partial \boldsymbol{K}}{\partial x_k} \approx \frac{\Delta \boldsymbol{K}}{\Delta x_k} = \frac{\boldsymbol{K}(x_k + \Delta x_k) - \boldsymbol{K}(x_k)}{\Delta x_k} \tag{3.23}$$

In the *Exact Semi-Analytical* (ESA) method (Olhoff et al. 1992) the derivatives $\partial \boldsymbol{K}/\partial x_k$ are computed by aggregating the contributions of every element, on the element level as follows

$$\frac{\partial \boldsymbol{k}}{\partial x_k} = \sum_{j=1}^{n_d} \frac{\partial \boldsymbol{k}}{\partial \alpha_j}\frac{\partial \alpha_j}{\partial x_k} \tag{3.24}$$

where n_d is the number of elemental nodal coordinates affected by the perturbation of the design variable x_k and a_j are the nodal coordinates of the element. The ESA method is more accurate than CSA and leads the mathematical optimizer to a faster convergence (Hinton and Sienz 1994).

Stress gradients can be calculated by differentiating the stress relationship as follows:

$$\boldsymbol{\sigma} = \boldsymbol{D}\cdot\boldsymbol{B}\cdot\boldsymbol{u} \quad \Rightarrow \tag{3.25}$$

$$\frac{\partial \boldsymbol{\sigma}}{\partial x_k} = \frac{\partial \boldsymbol{D}}{\partial x_k}\boldsymbol{B}\cdot\boldsymbol{u} + \boldsymbol{D}\frac{\partial \boldsymbol{B}}{\partial x_k}\boldsymbol{u} + \boldsymbol{D}\cdot\boldsymbol{B}\frac{\partial \boldsymbol{u}}{\partial x_k} \tag{3.26}$$

where $\boldsymbol{D}$ and $\boldsymbol{B}$ are the elasticity and the deformation matrices, respectively. In case that the elasticity matrix $\boldsymbol{D}$ is not a function of the design variables then Eq. (3.26) reduces to

$$\frac{\partial \boldsymbol{\sigma}}{\partial x_k} = \boldsymbol{D}\frac{\partial \boldsymbol{B}}{\partial x_k}\boldsymbol{u} + \boldsymbol{D}\boldsymbol{B}\frac{\partial \boldsymbol{u}}{\partial x_k} \tag{3.27}$$

In Eq. (3.27), $\partial \boldsymbol{u}/\partial x_k$ and $\partial \boldsymbol{B}/\partial x_k$ may be computed using a forward finite difference scheme. Using the values of $\partial \boldsymbol{\sigma}/\partial x_k$ the sensitivities of different types of stresses (e.g. the principal stresses or the equivalent stresses) can be readily calculated by analytically differentiating their expressions with respect to the design variables.

The numerical problem encountered in the sensitivity analysis phase when the global finite difference approach is implemented can be seen as an algebraic problem of the type $(\boldsymbol{K}+\boldsymbol{\Delta K}^i)\cdot\boldsymbol{u}^i=\boldsymbol{f}^i$ (i=1, ..., q).

The Global Finite Difference method

In this method the design sensitivities for the displacements $\partial \boldsymbol{u}/\partial x_k$ and the stresses $\partial \boldsymbol{\sigma}/\partial x_k$, which are needed for the gradients of the constraints, are computed using a forward difference scheme

$$\frac{\partial \boldsymbol{u}}{\partial x_k} \approx \frac{\Delta \boldsymbol{u}}{\Delta x_k} = \frac{\boldsymbol{u}(x_k + \Delta x_k) - \boldsymbol{u}(x_k)}{\Delta x_k} \tag{3.28}$$

$$\frac{\partial \boldsymbol{\sigma}}{\partial x_k} \approx \frac{\Delta \boldsymbol{\sigma}}{\Delta x_k} = \frac{\boldsymbol{\sigma}(x_k + \Delta x_k) - \boldsymbol{\sigma}(x_k)}{\Delta x_k} \tag{3.29}$$

The perturbed displacement vector $\boldsymbol{u}(x_k+\Delta x_k)$ of the finite element equations is evaluated by

$$\boldsymbol{K}(x_k + \Delta x_k) \cdot \boldsymbol{u}(x_k + \Delta x_k) = \boldsymbol{f}(x_k + \Delta x_k) \tag{3.30}$$

and the perturbed stresses $\boldsymbol{\sigma}(x_k + \Delta x_k)$ are computed from

$$\boldsymbol{\sigma}(x_k + \Delta x_k) = \boldsymbol{DB}(x_k + \Delta x_k) \cdot \boldsymbol{u}(x_k + \Delta x_k) \tag{3.31}$$

The GFD scheme is usually sensitive to the accuracy of the computed perturbed displacement vectors which is dependent on the magnitude of the perturbation of the design variables. Very small values of the perturbation will provide very small changes in the response of the structure and as a result the whole process can be affected by round-off errors of the computer. On the other hand, large values of the perturbation will result in decreased accuracy and truncation errors (Gill et al. 1981; Haftka and Gürdal 2007; Kibsgaard 1992), a typical behaviour of methods based on Taylor expansion with truncation. The magnitude of the perturbation is usually taken between 10^{-3} and 10^{-5} times the value of the design variable.

The numerical problem encountered in the sensitivity analysis phase when the Exact Semi-Analytical approach is used can be seen as an algebraic problem with multiple right-hand sides of the type $\boldsymbol{K}\cdot\boldsymbol{u}^i=\boldsymbol{f}^i$ (i=1, ..., q).

3.7 Evolutionary Algorithms (EAs)

Computer algorithms based on the process of natural evolution have been found capable to produce very powerful and robust search mechanisms although the similarity between these algorithms and the natural evolution is based on crude imitation of biological reality. The resulting *Evolutionary Algorithms* (EAs) are based on a population of individuals, each of which represents a search point in the space of potential solutions of a given problem. EAs constitute the most widely used class of methods of the probabilistic optimization category (Bäck and Schwefel 1993; Lagaros et al. 2002). These algorithms adopt a selection process based on the fitness of the individuals and some recombination operators. The best known EAs in this class include *Evolutionary Programming* (EP) (Fogel et al. 1966), *Genetic Algorithms* (GAs) (Goldberg 1989; Holland 1975) and *Evolution Strategies* (ES) (Rechenberg 1973; Schwefel 1981). The first attempt to use evolutionary algorithms took place in the sixties by a team of biologists (Barricelli 1962) and was focused in building a computer program that would simulate the process of evolution in nature. EAs have been found capable of producing very powerful and robust search mechanisms although the similarity between these algorithms and the natural evolution is based on crude imitation of biological reality. An EA for handling discrete optimization problems is an evolution-based procedure maintaining a population of potential solutions, which are subjected to processes of recombination/crossover, mutation and selection (Bäck and Schwefel 1993; Lagaros et al. 2002).

In structural optimization problems, where the objective function and the constraints are particularly highly non-linear functions of the design variables, the computational effort spent in gradient calculations required by the mathematical programming algorithms is usually large. Due to their random search, EAs proceed toward the optimal solution with slower rate than gradient-based optimizers and usually need a greater number of analyses. However, these analyses are less time consuming than the corresponding computations in MP algorithms, since EAs do not require expensive gradient calculations, and exhibit satisfactory convergence characteristics. In two studies by Papadrakakis et al. (1998b; 1999) it was found that probabilistic search algorithms are computationally efficient even if greater number of analyses is needed to reach the optimum, compared to MP methods.

Furthermore, EAs were found, due to their random search, to be less vulnerable to local optima, whereas mathematical programming algorithms may be easily trapped in local optima. EAs are therefore more robust and reliable in obtaining (or at least approaching) the global optimum for non-convex constrained optimization problems. Probabilistic optimization techniques are more robust and present a better global behavior than MP methods when confronted with complex optimization problems (Lagaros et al. 2002; Papadrakakis et al. 1999).

Both GAs and ES imitate biological evolution in nature and have three characteristics that make them differ from other conventional optimization algorithms:

i. In place of the usual deterministic operators, they use randomized operators: mutation, selection and recombination.

ii. Instead of a single design point, they work simultaneously with a population of design points in the space of design variables. This characteristic allows for a natural implementation on parallel computing environments (Adeli and Cheng 1994; Papadrakakis et al. 1998a; Thierauf and Cai 1995) which can significantly reduce the computational cost of the methodology.

iii. They can handle, with minor modifications continuous, discrete or mixed optimization problems.

GAs and ES are evolutionary methods that have been employed in a wide variety of structural engineering applications with very satisfactory results (Papadrakakis et al. 1998a; Papadrakakis et al. 1999). Evolutionary methods usually exhibit a fast rate of convergence towards the area of the global optimum. But after a while, convergence becomes slower, requiring comparatively many steps in order to converge. In any case, the computational cost of every step is rather small, as there is no need to compute gradients. In an attempt to improve performance, hybrid algorithms have been proposed that combine evolutionary type of algorithms with mathematical methods.

Some differences between GAs and ES stem from the numerical representation of the design variables used by these two algorithms. The basic GA operate on fixed-sized bit strings which are mapped to the values of the design variables, ES work on real-valued vectors. Another difference can be found in the use of the genetic operators. Although, both GA and ES use the mutation and recombination (crossover) operators, the role of these genetic operators is different. In GA mutation only serves to recover lost alleles, while in ES mutation implements some kind of hill-climbing search procedure with self-adapting step sizes.

In both algorithms recombination serves to enlarge the diversity of the population, and thus the covered search space. GAs' basic assumption is that the optimal solution can be found by assembling building blocks, i.e. partial pieces of solutions, while ES simply ensure the emergence of the best solutions. The most important consequence of this different approach is related to the recombination operator, viewed as essential for GAs and as potentially useful for ES. There is also a difference in treating constrained optimization problems where in the case of ES the death penalty method is always used, while in the case of GA only the augmented Lagrangian method can guarantee the convergence to a feasible solution.

The modern tendency seems to follow combinations of the two approaches, since GAs users have turned to real number representations when dealing with real numbers following experimental results or heuristic demonstrations, whereas ES users have included recombination as a standard operator and have designed special operators for non real-valued problems (Schoenauer 1995). ES appear to be more robust and are likely to

achieve a higher rate of convergence than GAs, due to their self-adaptation search mechanism, in solving real-world problems (Hoffmeister and Back 1991).

3.8 Genetic Algorithms (GAs)

The most widely used type of EAs is the Genetic Algorithm method which represents a model of machine learning that uses a genetic metaphor (Goldberg 1989). Implementations of this model typically use fixed-length character strings (binary or real valued) to represent their genetic information, together with a population of individuals which undergo mutation and crossover in order to guide the search process towards the optimum in the search space. A string, which represents a member of the genetic population, is sometimes referred to as a *genotype* or, alternatively, as a *chromosome*.

The basic GA algorithm for structural optimization applications is shown in Figure 3.5.

1. *Initialization step:* Random generation of an initial population of vectors of the design variables $\boldsymbol{x}^j$ ($j=1,\ldots,n_{\text{pop}}$) which are encoded in binary strings.
2. *Analysis step*: Solve the structural analysis problem $\boldsymbol{K}(\boldsymbol{x}^j)\cdot\boldsymbol{u}^j=\boldsymbol{f}$, ($j=1,\ldots,n_{\text{pop}}$).
3. *Fitness evaluation step*: Each member of the population is evaluated by computing the representative penalized objective and the corresponding fitness functions, using an appropriate penalty function.
4. *Selection step*: Selection operator is applied to the current population to create an intermediate one.
5. *Generation step*: In order to create the next generation, crossover and mutation operators are applied to the intermediate population to create the next population $\boldsymbol{t}^j$ ($j=1,\ldots,n_{\text{pop}}$).
6. *Analysis-Fitness evaluation step*: Solve $\boldsymbol{K}(\boldsymbol{t}^j)\cdot\boldsymbol{u}^j=\boldsymbol{f}$, ($j=1,\ldots,n_{\text{pop}}$).
7. *Convergence check*: If satisfied stop, else go to *step 4*.

Figure 3.5 Basic GA algorithm for structural optimization.

The size of the population that is suggested in the literature (Goldberg 1989) can be determined using the following equation:

$$n_{pop} = 2^k \cdot (\zeta / k) \tag{3.32}$$

where ζ is the length of the binary string and k is the average size of the schema. Let us consider an optimization problem with three design variables. The first two design variables are decoded using 4 binary digits while the third one using 5. The length of the binary string is $\zeta=4+4+5=13$, the average length of the schema is $k=13/3\approx4.33$ and the

population size according to Eq. (3.32) should be equal to 60. Sometimes Eq. (3.32) gives a large population size which makes the optimization step computationally too expensive.

3.8.1 Encoding

To apply the GA method in its basic form, a suitable encoding of the real valued vectors $\boldsymbol{x}^j$ (j=1,...,n_{pop}) as a binary string is required. The real valued form of a design vector is called *phenotype* while the binary one is called *genotype*. This is achieved by subdividing the binary string into n (number of design variables) segments, decoding each segment to yield the corresponding real value using the binary representation. For example, if there are n design variables in an optimization problem and each design variable is encoded as an *L-digit* binary sequence, then the length of the string is of ζ=n·L binary digits. In the case of discrete design variables each discrete value is assigned to a binary string, while in the case of continuous design variables the design space is divided into a number of intervals (a power of 2).

3.8.2 Fitness function

The fitness function value is assigned to each member of the population and is used as a measure of the suitability of each member compared to the others. Evaluation of a string refers to the evaluation of the objective function value of that string and it is independent of the evaluation of any other string of the population. The fitness of that string, however, is always defined with respect to other members of the current population. The fitness is used to determine the selection probability of this chromosome to become the parent chromosome for the generation of the new chromosomes. In the basic genetic algorithm, fitness is defined by: $F_i'/\overline{F}'$ where F_i' is the penalized objective function associated with the i-th string, while $\overline{F}'$ is the average penalized objective function value of all the strings in the population. The evaluation of the penalized objective function is described in detail in Section 0 (Constraint handling techniques). Fitness can also be assigned based on a string's rank in the population or by sampling methods, such as tournament selection.

3.8.3 Selection

A number of ways have been proposed to perform the selection, among which the most widely used are the *Tournament selection*, *Roulette Wheel selection* and the *Ranking selection* schemes. According to the Tournament selection scheme, each member of the intermediate population is selected to be the best member from a randomly selected group of members belonging to the current population. According to the Roulette Wheel selection scheme, the population is laid out in random order as in a pie graph, where for each individual a space on the pie graph is assigned in proportion to its fitness. Next an outer roulette wheel is placed around the pie with n_{pop} equally spaced pointers. A single

spin of the roulette wheel will now simultaneously pick all n_{pop} members of the intermediate population. According to the Ranking selection scheme, the members of the population are classified based on the fitness function value, they are assigned with a classification rank and their selection is based on this rank. The tournament selection scheme has been found to be more efficient (Goldberg and Deb 1991). Elitism is a technique that is used in conjunction with the selection operator. According to this technique the best chromosome is directly copied to the new population avoiding the genetic operators. Elitism often increases the performance of the algorithm, by preventing loss of the best-found solution.

3.8.4 Genetic Operators

Crossover and mutation are the basic genetic operators. Crossover is a reproduction operator, which forms a new chromosome by combining parts of two "parent" chromosomes. Many types of crossover schemes have been proposed, such as: (i) one point; (ii) two-point; (iii) multi-point and (iv) uniform crossover. Single and multi-point crossover schemes define cross points as places between loci where a chromosome can be split. Uniform crossover generalizes this scheme by randomly creating a crossover mask, having the same length as the parent chromosomes. This mask defines which parent will contribute its corresponding bit to the offspring chromosome. When a population becomes quite homogeneous, another factor becomes important: whether the offspring produced by crossover will be different than their parents. The uniform crossover operator has been proved to be able to produce such diversity (Spears and De Jong 1990).

Mutation is a reproduction operator, which forms a new chromosome by making (usually small) alterations to the values of genes in a copy of a single parent chromosome and serves only to recover lost alleles, where allele is the value of a gene. For binary encoding each gene may have an allele of 0 or 1. The main purpose of the mutation operator is to maintain diversity within the population and inhibit premature convergence.

3.9 Evolution Strategies (ES)

Evolution Strategies (ES) were proposed for parameter optimization problems in the seventies by Rechenberg (1973) and Schwefel (1981). Similar to GAs, Evolution Strategies imitate biological evolution in nature. ES were initially applied to continuous optimization problems, but after a while they were also implemented in discrete and mixed optimization problems (Thierauf and Cai 1995; Thierauf and Cai 1996). ES have been also applied for structural optimization problems under dynamic loading and especially under earthquake loading within a certain seismic code (Cai 1995; Lagaros et al. 2005b; Papadrakakis et al. 2000a; Papadrakakis et al. 2000b; Papadrakakis et al. 2001a; Papadrakakis et al. 2002a; Papadrakakis et al. 2002b; Plevris et al. 2007).

3.9.1 ES for continuous optimization problems

The ES methods can be divided into: (i) the simple two-membered Evolution Strategy (2-ES); and (ii) the multi-membered Evolution Strategy (M-ES).

The two-membered ES

The earliest ES were based on a population consisting of one individual only which produced only one offspring. This two-membered scheme is the minimal concept for an imitation of organic evolution. The two principles of mutation and selection, which Darwin (1859) recognized to be most important, are taken as rules for variation of the parameters and for recursion of the iteration sequence respectively.

The two-membered ES for the solution of the optimization problem works in two steps:

Step 1 (*Mutation*): The parent $\boldsymbol{x}^{p(g)}$ of the generation g produces an offspring $\boldsymbol{x}^{o(g)}$, whose genotype is slightly different from that of the parent

$$\boldsymbol{x}^{o(g)} = \boldsymbol{x}^{p(g)} + \boldsymbol{z}^{(g)} \tag{3.33}$$

where

$$\boldsymbol{z}^{(g)} = \left[z_1^{(g)}, \ldots, z_n^{(g)} \right]^{\mathrm{T}} \tag{3.34}$$

is a random vector.

Step 2 (Selection): The selection chooses the best individual between the parent and the offspring to survive

$$\boldsymbol{x}^{p(g+1)} = \begin{cases} \boldsymbol{x}^{o(g)} & \text{if} \quad \text{design } \boldsymbol{x}^{o(g)} \text{ is feasible and } f(\boldsymbol{x}^{o(g)}) \le f(\boldsymbol{x}^{p(g)}) \\ \boldsymbol{x}^{p(g)} & \text{otherwise} \end{cases} \tag{3.35}$$

The question on how to choose the random vector $\boldsymbol{z}^{(g)}$ in *Step 1* is very important. This choice has the role of mutation. Mutation can be understood as random, purposeless events, which occur very rarely, therefore it is a logical choice to use a probability distribution according to which small changes occur frequently, but large ones only rarely. Two requirements arise by analogy with natural evolution:

i. the expected mean value ξ_i for a component $z_i^{(g)}$ to be zero;

ii. the variance σ_i^2, the average squared standard deviation from mean value, to be small.

The probability density function for normally distributed random events is given by

$$p\left(z_i^{(g)}\right) = \frac{1}{\sigma_i \sqrt{(2\pi)}} \exp\left(-\frac{(z_i^{(g)} - \xi_i)^2}{2\sigma_i^2} \right) \tag{3.36}$$

When ξ_i=0 the so-called (0, σ_i) normal distribution is obtained. By analogy with other deterministic search strategies, σ_i can be seen as the step length, in the sense that it represents average values of the length of the random steps. If the step length is too small the search takes an unnecessarily large number of iterations. On the other hand, if the step length is too large, the optimum can only be crudely approached and the optimization search can even get stuck far away from the global optimum. Thus, as in all optimization strategies, the step length control is one of the most important parts of the algorithm after the recursion formula, and it is further more closely linked to the convergence behavior.

Multi-membered ES

The multi-membered Evolution Strategy differs from the two-membered strategy in the size of the population. In M-ES a population of μ parents will produce λ offspring. Thus the two steps of the procedure are defined as follows:

Step 1 (*recombination and mutation*): The population of μ parents at g-th generation produces λ offspring. The genotype of any descendant differs only slightly from that of its parents.

Step 2 (*selection*): There are two different types of selection for the multi-membered ES:

(μ+λ)-ES: The best μ individuals are selected from a temporary population of (μ+λ) individuals to form the parents of the next generation.

(μ, λ)-ES: The μ individuals produce λ offspring ($\mu < \lambda$) and the selection process defines a new population of μ individuals from the set of λ offspring only. The parents are not taken into account in the selection process for the new generation and as a result an individual cannot "live" for more than one generation.

In the second type, the existence of each individual is limited to one generation. This allows the (μ,λ)-ES selection to perform better on problems with an optimum moving over time, or on problems where the objective function is noisy.

In *Step 1*, for every offspring vector a temporary parent vector $\tilde{\boldsymbol{x}} = [\tilde{x}_1,...,\tilde{x}_n]^{\mathrm{T}}$ is first built by means of recombination. For continuous problems the following recombination cases can be used

$$\tilde{x}_i = \begin{cases} x_i^a \text{ or } x_i^b \text{ randomly} & \text{(A)} \\ 1/2\left(x_i^a + x_i^b\right) & \text{(B)} \\ x_i^{\mathrm{rand}} & \text{(C)} \\ x_i^a \text{ or } x_i^{\mathrm{rand}} \text{ randomly} & \text{(D)} \\ 1/2\left(x_i^a + x_i^{\mathrm{rand}}\right) & \text{(E)} \end{cases} \tag{3.37}$$

where $\tilde{x}_i$ is the i-th component of the temporary parent vector $\tilde{\boldsymbol{x}}$, $x^{\alpha}{}_i$ and $x^{b}{}_i$ are the i-th components of the vectors $\boldsymbol{x}^{\alpha}$ and $\boldsymbol{x}^{b}$ which are two parent vectors randomly chosen from

the population. In case C of Eq. (3.37), $\tilde{x}_i = x_i^{\text{rand}}$ means that the i-th component of $\tilde{\boldsymbol{x}}$ is chosen randomly from the i-th components of all μ parent vectors. From the temporary parent $\tilde{\boldsymbol{x}}$ an offspring can be created in the same way as in the case of two-membered ES with Eq. (3.33).

Termination criteria

For continuous optimization problems with the Multi-membered ES formulation, the procedure terminates when one of the following termination criteria is satisfied:

i. The absolute or relative difference between the best and the worst objective function values is less than a given threshold value ε_1;

ii. The mean value of the objective values from all parent vectors in the last $2 \cdot n$ generations has not been improved by less than a given threshold value ε_2.

3.9.2 ES for discrete optimization problems

In engineering practice the design variables are usually not continuous because the structural parts are constructed with certain variation of their dimensions. Thus design variables can only take values from a predefined discrete set. For the solution of discrete optimization problems Thierauf and Cai (1996) have proposed a modified ES algorithm. The basic differences between discrete and continuous ES are focused on the mutation and the recombination operators. The multi membered ES (M-ES) adopted in the present thesis uses three operators: (i) recombination; (ii) mutation; and (iii) selection operators that can be included in the algorithm as follows:

Step 1 (recombination and mutation)

The population of μ parents at g-th generation produces λ offspring. The genotype of any descendant differs only slightly from that of its parents. For every offspring vector a temporary parent vector $\tilde{\boldsymbol{x}} = [\tilde{x}_1, \ldots, \tilde{x}_n]^{\mathrm{T}}$ is first built by means of recombination. For discrete problems the following recombination cases can be used

$$\tilde{x}_i = \begin{cases} x_i^a \text{ or } x_i^b \text{ randomly} & \text{(A)} \\ x_i^{\text{best}} \text{ or } x_i^b \text{ randomly} & \text{(B)} \\ x_i^{\text{rand}} & \text{(C)} \\ x_i^a \text{ or } x_i^{\text{rand}} \text{ randomly} & \text{(D)} \\ x_i^{\text{best}} \text{ or } x_i^{\text{rand}} \text{ randomly} & \text{(E)} \end{cases} \tag{3.38}$$

$\tilde{x}_i$ is the i-th component of the temporary parent vector $\tilde{\boldsymbol{x}}$, $x^{\alpha}{}_i$ and $x^{b}{}_i$ are the i-th components of the vectors $\boldsymbol{x}^{\alpha}$ and $\boldsymbol{x}^{b}$, which are two parent vectors randomly chosen from the population. The vector $\boldsymbol{x}^{\text{best}}$ is not randomly chosen but is the best of the μ parent vectors in the current generation. In case C of Eq. (3.38), $\tilde{x}_i = x_i^{\text{rand}}$ means that the i-th compo-

nent of $\tilde{\boldsymbol{x}}$ is chosen randomly from the i-th components of all μ parent vectors. From the temporary parent $\tilde{\boldsymbol{x}}$ an offspring can be created following the mutation operator.

Let as consider the temporary parent $\boldsymbol{x}^{p(g)}$ of the generation g that produces an offspring $\boldsymbol{x}^{o(g)}$ through the mutation operator as follows

$$\boldsymbol{x}^{o(g)} = \boldsymbol{x}^{p(g)} + \boldsymbol{z}^{(g)} \tag{3.39}$$

where

$$\boldsymbol{z}^{(g)} = \left[z_1^{(g)}, \ldots, z_n^{(g)} \right]^{\mathrm{T}} \tag{3.40}$$

is a random vector. Mutation can be understood as random, purposeless events, which occur very rarely. If one interprets them as a set of many individual events, the "natural" choice would be to use a probability distribution according to which small changes occur frequently, but large ones only rarely. As a result of this assumption two requirements arise together by analogy with natural evolution:

i. the expected mean value ξ_i for a component $z_i^{(g)}$ to be zero;

ii. the variance σ_i^2, the average squared standard deviation from mean value, to be small.

The mutation operator in the continuous version of ES produces a normally distributed random change vector $\mathbf{z}^{(g)}$. Each component of this vector has small standard deviation value σ_i and zero mean value. As a result of this, there is a possibility that all components of a parent vector will be changed, but usually the changes are small. In the discrete version of ES the random vector $\mathbf{z}^{(g)}$ is properly generated in order to force the offspring vector to move to another set of discrete values. The fact that the difference between any two adjacent values can be relatively large is against the requirement that the variance σ_i^2 be small. For this reason it is suggested (Thierauf and Cai 1996) that not all the components of a parent vector, but only a few of them (e.g. ℓ), should be randomly changed in every generation. This means that n-ℓ components of the randomly changed vector $\boldsymbol{z}^{(g)}$ will have zero value. In other words, the terms of vector $\mathbf{z}^{(g)}$ are derived from

$$z_i^{(g)} = \begin{cases} (\kappa + 1) \cdot \delta x_i & \text{for } \ell \text{ randomly chosen components} \\ 0 & \text{for } n\text{-}\ell \text{ other components} \end{cases} \tag{3.41}$$

where δx_i is the difference between two adjacent values in the discrete set and κ is a random integer number, which follows the Poisson distribution

$$p(\kappa) = \frac{(\gamma)^{\kappa}}{\gamma!} e^{-\gamma} \tag{3.42}$$

where γ is the standard deviation as well as the mean value of the random number κ. For a very small γ (e.g. 0.001) the probability that κ will be zero is $p(0) \approx 99.9\%$. For greater values of γ the probability that κ be zero increases (e.g. for $\gamma = 0.01$, $p(0) \approx 99\%$, for $\gamma = 0.1$,

$p(0) \approx 90.5\%$). Figure 3.6 shows the discrete Poisson distribution for three values of the parameter γ.

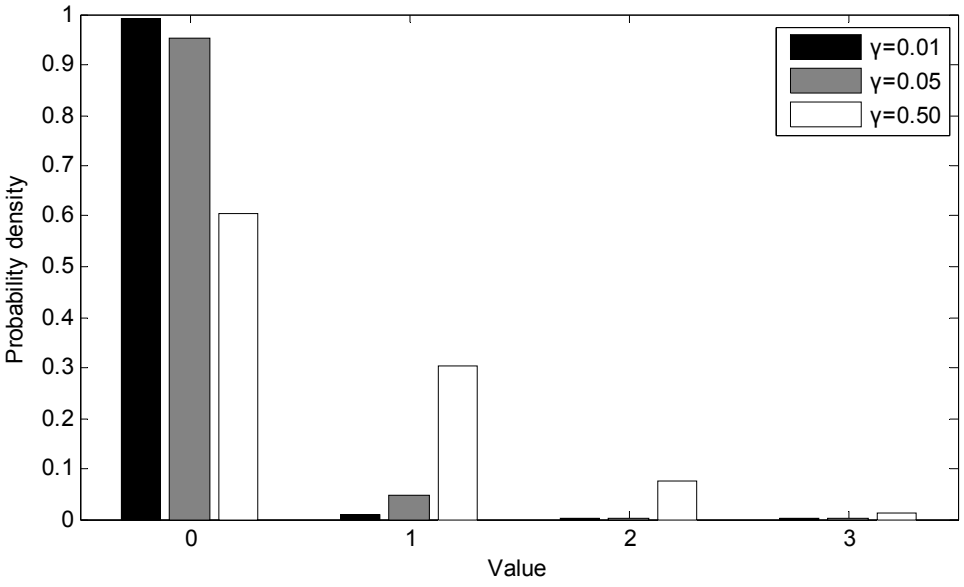


Figure 3.6 Discrete Poisson distributions for various values of the parameter γ.

The choice of ℓ depends on the size of the problem. It is usually taken as the 1/5 of the total number of design variables. The ℓ components are selected using uniform random distribution in every generation according to Eq. (3.41).

Step 2 (selection)

In the discrete version of ES, there are again two different selection schemes: (i) $(\mu+\lambda)$-ES; and (ii) (μ,λ)-ES, as was described in detail in Section 3.9.1 for the Multi-membered ES case.

Termination criteria

For discrete optimization the procedure terminates when one of the following termination criteria is satisfied:

1. When the best value of the objective function in the last $4n \cdot \mu/\lambda$ generations remains unchanged.
2. When the mean value of the objective values from all parent vectors in the last $2n \cdot \mu/\lambda$ generations has not been improved by less than a threshold value ε_b, e.g. $\varepsilon_b = 0.0001$.
3. When the relative difference between the best objective function value and the mean value of the objective function values from all parent vectors in the current generation is less than a threshold value ε_c, e.g. $\varepsilon_c = 0.0001$.

4. When the ratio μ_b/μ has reached a given value ε_d, e.g. ε_d=0.5 to 0.8, where μ_b is the number of the parent vectors in the current generation with the best objective function value.

3.9.3 ES in structural optimization problems

The ES optimization procedure starts with a set of parent vectors. If any of these parent vectors gives an infeasible design then this parent vector is modified until it becomes feasible. Subsequently, the offspring are generated and checked if they are in the feasible region. According to $(\mu+\lambda)$ selection scheme in every generation the values of the objective function of the parent and the offspring vectors are compared and the worst vectors are rejected, while the remaining ones are considered to be the parent vectors of the new generation. On the other hand, according to (μ,λ) selection scheme only the offspring vectors of each generation are used to produce the new generation. This procedure is repeated until the chosen termination criterion is satisfied.

The computational efficiency of the multi-membered ES is affected by the number of parents and offsprings involved. It has been observed that values of μ and λ should be close to the number of the design variables produce best results (Papadrakakis et al. 1999). The ES algorithm for structural optimization applications is shown in Figure 3.7.

1. *Selection step*: Selection of μ parent vectors of the design variables.
2. *Analysis step*: Solution of the FE problems corresponding to the parent vectors, $\boldsymbol{K}(\boldsymbol{x}^i)\cdot\boldsymbol{u}^i=\boldsymbol{f}$, $(i=1,\ldots,\mu)$, where $\boldsymbol{K}$ is the stiffness matrix of the structure and $\boldsymbol{f}$ is the loading vector.
3. *Constraints check*: Handling of the constraints for the parent vectors. If the death penalty method is used, all parent vectors become feasible.
4. *Offspring generation*: Generation of the λ offspring vectors of the design variables using recombination/crossover and mutation operators.
5. *Analysis step*: Solution of the FE problems corresponding to the generated offspring vectors, $\boldsymbol{K}(\boldsymbol{x}^j)\cdot\boldsymbol{u}^j=\boldsymbol{f}$, $(j=1,\ldots,\lambda)$.
6. *Constraints check*: Handling of the constraints for the offspring vectors. If the death penalty method is used, then if constraints are satisfied continue, else change the offspring and go to *step 4*.
7. *Parents' selection step*: Selection of the next generation parents according to $(\mu+\lambda)$ or (μ,λ) selection schemes.
8. *Convergence check*: If satisfied stop, else go to *step 3*.

Figure 3.7 ES algorithm for structural optimization.

The robustness of the above EA procedure in locating optimum structural designs has been studied in terms of various parameters of the optimizer (μ and λ-values, initial design, convergence criterion, initial seed of the random number generator, etc.) in a number of studies (Lagaros et al. 2005a; Lagaros et al. 2004; Papadrakakis et al. 2003).

Constraint handling techniques

Various methods have been proposed for handling non-linear constraints by Evolutionary Algorithms in general. Koziel and Michalewicz (1999) grouped them into four categories:

1. Methods based on preserving feasibility of the solutions;
2. Methods based on penalty functions;
3. Methods that search for feasibility;
4. Other hybrid methods.

The constrained handling techniques for ES used in the present thesis are: (i) the death penalty approach, belonging to the first category of the methods described above; and (ii) the penalty function approach, belonging to the second category. Details on constraint handling techniques can be found in Section 3.11.4.

3.10 Cascade Evolutionary Algorithm (CEA)

It is generally accepted that there is no unique optimization algorithm capable of handling effectively all existing optimization problems (Wolpert and Macready 1997). This has been also demonstrated by Patnaik et al. (1996), who have considered a number of optimizers to solve several problems included in a test-bed. They concluded that any individual optimizer succeeded in solving a number of the test-bed problems, but not all of them, although every one of these problems could be successfully solved by at least one of the optimizers.

According to the *Cascade Optimization* strategy, a remedy to this situation can be a multi-stage procedure in which various optimizers are implemented in a successive manner (Papadrakakis et al. 1999; Patnaik et al. 1997). In the first stage of the cascade procedure the initial optimizer starts from a user specified design known as 'cold-start'. The intermediate optimal solution reached in the first cascade stage, which may be perturbed using a pseudo-random technique, is called a 'hot-start' and is used to initiate the second optimization stage. Accordingly, each optimization stage of the cascade procedure starts from the (possibly perturbed) optimum design achieved in the previous stage. Thus, each cascade stage after the first stage initiates from a hot-start and produces a new hot-start for the next stage coupling this way the autonomous computations of successive optimization stages.

In general, the optimization algorithms implemented in each stage of a cascade process may vary. Cascade optimization has been implemented using different deterministic op-

timizers in the cascade stages (Patnaik et al. 1997), using both deterministic and probabilistic optimizers one after the other (Papadrakakis et al. 1999) as well as using probabilistic optimizers (Charmpis et al. 2005). The main argument in combining different optimizers in a successive manner has to do with maximizing the exploitation of the advantages offered by various optimization algorithms and diminishing the influence of the corresponding disadvantages on the optimum design achieved. The selection of the optimizers to be included in a cascade procedure, their exact sequence and the number of cascade stages performed can be determined via a trial-and-error process.

In case of discrete optimization problems, the cascade process can also offer another advantage, as differentiated search paths can also be obtained by utilizing a different database containing the cross-section sizes of the structural members, in each cascade stage. According to this approach, during the first stage of the cascade process, a coarse database can be used, while for the next stages the database is enriched until a full database is used for the final stage. The great difficulties that can be encountered when attempting to handle large databases necessitate the use of relatively small databases instead (e.g. (Greiner et al. 2004; Sarma and Adeli 2002)). This restriction in the size of the database aims in constructing a design space that is better manageable and therefore more effectively searchable by standard optimization schemes, which are most likely to become confused and ineffective when confronted with a large database.

In the present thesis, the idea of cascading is implemented in an Evolutionary Algorithm context. The resulting *Cascade Evolutionary Algorithm* (CEA) is designated by CEA(μ+λ) and consists of a number of optimization stages, each of which employs the same EA optimizer. In order to differentiate the search paths followed by the same optimization algorithm during the cascade stages, the initial conditions of individual optimization runs are suitably controlled by using in each stage: (i) a different initial design (each stage initiates from the solution of the previous stage); and (ii) a different seed for the random number generator of the EA process. The cascade procedure is terminated when no more improvement on the optimum solution vector can be attained over a small number of optimization stages.

3.11 Particle Swarm Optimization (PSO)

3.11.1 Introduction

Many probabilistic-based search algorithms have been inspired by natural phenomena, such as Evolutionary Programming, Genetic Algorithms, Evolution Strategies, among others. Recently, a family of optimization methods has been developed based on the simulation of social interactions among members of a specific species looking for food or resources in general. One of these methods is the *Particle Swarm Optimization* (PSO) method that is based on the behavior reflected in flocks of birds, bees and fish that adjust their physical movements to avoid predators and seek for food. The method has

been given considerable attention in recent years among the optimization research community.

A swarm of birds or insects or a school of fish searches for food, resources or protection in a very typical manner. If a member of the swarm discovers a desirable path to go, the rest of the swarm will follow quickly. Every member searches for the best in its locality, learns from its own experience as well as from the others typically from the best performer among them. Even human beings show a tendency to behave in this way as they learn from their own experience, their immediate neighbors and the ideal performers in the society. The PSO method mimics the behavior described above. The algorithm was first proposed by Kennedy and Eberhart (1995). It is a population-based optimization method built on the premise that social sharing of information among the individuals can provide an evolutionary advantage.

PSO has been found to be highly competitive for solving a wide variety of optimization problems (Bochenek and Foryś 2006; He and Wang 2007; Liang and Suganthan 2006; Mezura-Montes and Lopez-Ramirez 2007; Munoz-Zavala et al. 2006; Perez and Behdinan 2007b; Ye et al. 2007). A number of advantages over other algorithms make PSO a prospective candidate to be used also in structural optimization problems. It can handle non-linear, non-convex design spaces with discontinuities. Compared to other non-deterministic optimization methods it is considered efficient in terms of number of function evaluations as well as robust since it usually leads to better or the same quality of results. Its easiness of implementation makes it more attractive as it does not require specific domain knowledge information, while being a population-based algorithm, it can be straight forward implemented in parallel computing environments leading to a significant reduction of the total computational cost. Compared to GA, PSO is easier to implement and there are only a few parameters to adjust. PSO has been successfully applied to many fields, such as mathematical function optimization, artificial neural network training and fuzzy system control.

In spite of the popularity of PSO as an efficient optimizer, it has been until relatively recently that its use has focused more on engineering optimization problems, mainly because of the lack of constraint-handling techniques explicitly designed to be coupled with a PSO algorithm. Promising results have been presented in the areas of structural shape optimization (Fourie and Groenwold 2002; Venter and Sobieszczanski-Sobieski 2004) as well as topology optimization (Fourie and Groenwold 2001). Perez and Behdinan (2007a; 2007b) implemented the PSO algorithm for constrained structural optimization of plane and space truss structures while Li et al. (2007) tried a heuristic PSO scheme for the optimization of truss structures.

3.11.2 Relationship of PSO with Evolutionary Algorithms

PSO shares many similarities with evolutionary computation techniques, such as Genetic Algorithms, but the conceptual difference lies in its definition which is given in a social rather than a biological context. The common features of the two optimization ap-

proaches include the population concept of the design vectors, initialization with a population of random solutions, a fitness value to evaluate performance, searching for optima by updating iterations (generations) based on a stochastic process, no requirement for gradient information or user-defined initial estimates and no guaranteed final success. However, unlike GA, PSO has no genetic operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following a velocity update rule. The information sharing mechanism in PSO is significantly different compared to GA. In GA chromosomes share information with each other, so the whole population moves like one group towards an optimal area. In PSO, only Gbest (the global best particle) communicates the information to the others, forming a one-way information sharing mechanism. Compared to Genetic Algorithms, according to the study of Hassan et al. (2005), PSO and GA can both obtain high quality solutions, yet the computational effort required by PSO to arrive to such high quality solutions is less than the corresponding effort required by GA.

According to Angeline (1998), two main distinctions can be made between PSO and an evolutionary algorithm:

i. EAs rely on three mechanisms in their processing: parent representation, selection of individuals and the fine tuning of their parameters. In contrast, PSO only relies on two mechanisms, since PSO does not adopt an explicit selection function. The absence of a selection mechanism in PSO is compensated by the use of leaders to guide the search. However, there is no notion of offspring generation in PSO as with EAs.

ii. The manipulation of the individuals is different in EAs and PSO. PSO uses an operator that sets the velocity of a particle to a particular direction. This can be seen as a directional mutation operator in which the direction is defined by both the particle's personal best and the global best (of the swarm). If the direction of the personal best is similar to the direction of the global best, the angle of potential directions will be small, whereas a larger angle will provide a larger range of exploration. In contrast, EAs use a mutation operator that can set an individual in any direction (although the relative probabilities for each direction may be different). In fact, the limitations exhibited by the directional mutation of PSO has led to the use of mutation operators similar to those adopted in EAs.

3.11.3 The PSO algorithm for unconstrained optimization

In a PSO formulation, multiple candidate solutions coexist and collaborate simultaneously. Each solution is called a "particle" that has a position and a velocity in the multidimensional design space. A particle "flies" in the problem search space looking for the optimal position. As "time" passes through its quest, a particle adjusts its velocity and position according to its own "experience" as well as the experience of other (neighboring) particles. Particle's experience is built by tracking and memorizing the best position encountered. As every particle remembers the best position it has visited during its

"flight", the PSO possesses a memory. A PSO system combines local search method (through self experience) with global search method (through neighboring experience), attempting to balance exploration and exploitation.

Mathematical formulation of PSO

Each particle maintains two basic characteristics, velocity and position in the multi-dimensional search space, which are updated in a stochastic way as follows:

$$\boldsymbol{v}^j(t+1) = w\boldsymbol{v}^j(t) + c_1\boldsymbol{r}_1 \circ \left(\boldsymbol{x}^{\mathrm{Pb},j} - \boldsymbol{x}^j(t)\right) + c_2\boldsymbol{r}_2 \circ \left(\boldsymbol{x}^{\mathrm{Gb}} - \boldsymbol{x}^j(t)\right) \tag{3.43}$$

$$\boldsymbol{x}^j(t+1) = \boldsymbol{x}^j(t) + \boldsymbol{v}^j(t+1) \tag{3.44}$$

where $\boldsymbol{v}^j(t)$ denotes the velocity vector of particle j at time t, $\boldsymbol{x}^j(t)$ represents the position vector of particle j at time t, vector $\boldsymbol{x}^{\mathrm{Pb},j}$ is the memory of particle j at current iteration (the personal best ever position of the particle, corresponding to the objective function value $Pbest_j$), and vector $\boldsymbol{x}^{\mathrm{Gb}}$ is the global best location found by the entire swarm up to the current iteration (corresponding to the objective function value *Gbest*, the same for all particles). The acceleration coefficients c_1 and c_2 represent "trust" settings which indicate the degree of confidence in the best solution found by the individual particle (c_1 - cognitive parameter) and by the whole swarm (c_2 - social parameter), respectively, while $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are two vectors containing random numbers with uniform distribution in the interval [0, 1].

The symbol "$\circ$" of Eq. (3.43) denotes the Hadamard product (entry-wise vector or matrix multiplication). This notation is used together with vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, as different random numbers have to be applied for every dimension of a particle at every iteration. If scalar values r_1 and r_2 were used for the random numbers multiplication of Eq. (3.43), then the same random value would multiply every dimension of a particle, which is not desirable in PSO.

Figure 3.8 shows a visualization of a particle's movement, in a two-dimensional design space, according to Eqs. (3.43) and (3.44). The particle's current position $\boldsymbol{x}^j(t)$ at time t is represented by the dotted circle at the lower left of the drawing, while the new position $\boldsymbol{x}^j(t+1)$ at time $t+1$ is represented by the dotted bold circle at the upper right of the drawing. It can be seen how the particle's movement is affected by:

i. The previous velocity $\boldsymbol{v}^j(t)$;

ii. The personal best ever position of the particle, $\boldsymbol{x}^{\mathrm{Pb},j}$, at the right of the figure;

iii. The global best location found by the entire swarm, $\boldsymbol{x}^{\mathrm{Gb}}$, at the upper left of the figure.

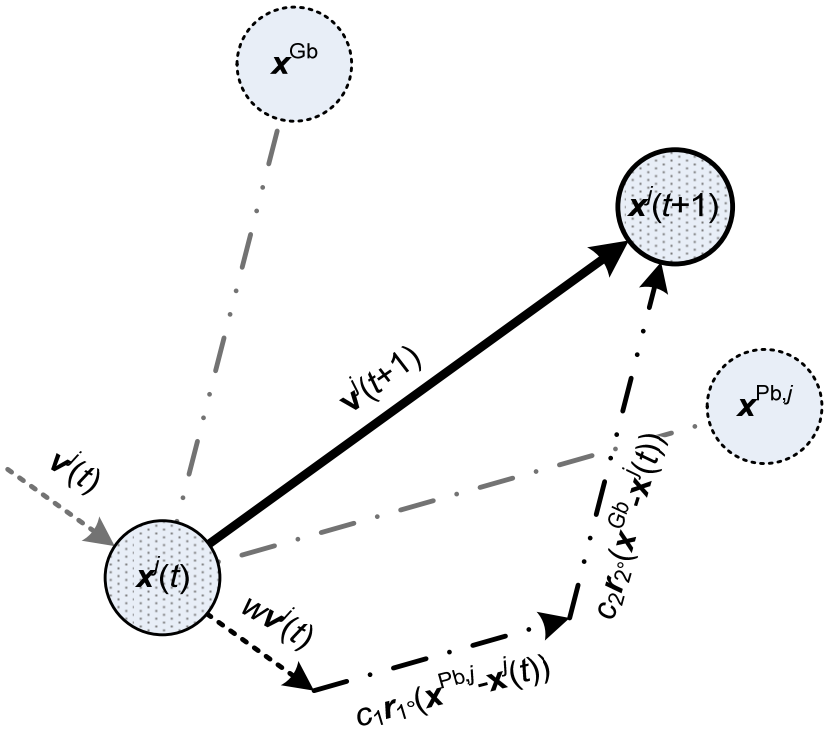


Figure 3.8 Visualization of the particle's movement in a two-dimensional design space.

The neighborhood of a particle includes a number of other particles with which a particle shares information. In the original PSO, two different kinds of neighborhoods were defined (Kennedy and Mendes 2002):

i. The *Gbest* swarm, shown in Figure 3.9(a), where all the particles are neighbors of each other; thus, the position of the best overall particle in the swarm is used in the social term of the velocity update equation. It is assumed that Gbest swarms converge fast, as all the particles are attracted simultaneously to the best part of the search space. However, if the global optimum is not close to the best particle, it may be impossible to the swarm to explore other areas also; this means that the swarm can be trapped in local optima.

ii. The *Lbest* swarm, shown in Figure 3.9(b), where only a specific number of particles (neighbor count) can affect the velocity of a given particle. The swarm will converge slower but may have larger possibility to locate the global optimum. Instead of the global best location found by the entire swarm, a local best location of each particle's neighborhood is used. Thus, information is shared only among members of the same neighborhood.

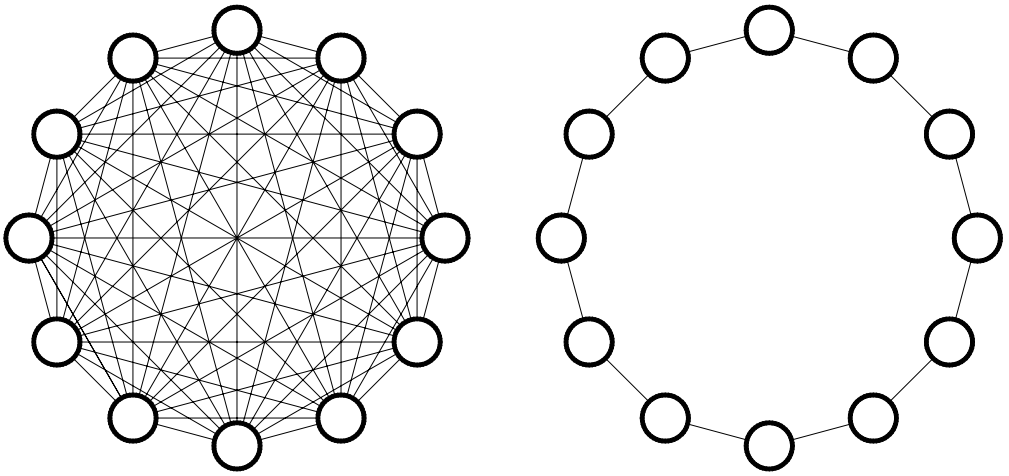


Figure 3.9 Graphical representations of the two topologies: (a) Gbest, (b) Lbest with two neighbors for each particle.

It has been shown that the flow of information through social networks is affected by several aspects of the networks (Watts 1999; Watts and Strogatz 1998). The first measure is the degree of connectivity among nodes in the net. Each individual in a particle swarm identifies the best point found by its *k* neighbors; *k*, then, is the variable that distinguishes Lbest from Gbest topologies, and is likely to affect performance.

The traditional particle swarm topology, Gbest, instantiates the most immediate communication possible as all particles are directly connected to the best solution in the population. On the other hand, the ring lattice topology Lbest is the slowest, most indirect communication pattern. Where a particle *j* is opposite a particle *z* on the lattice, a good solution found by *j* has to pass through *j*'s immediate neighbor, that particle's immediate neighbor, and so on, until it reaches particle *z*. Thus a solution found by *j* moves very slowly around the ring. Both Gbest and Lbest can be seen as "social" neighborhoods, as the relations among particles do not depend on their positions in the search space, but on "external" relationships that are not dependent on the problem that is being solved.

In the present thesis, the formulation of Eq. (3.43), the global best location found by the entire swarm up to the current iteration ($\boldsymbol{x}^{\mathrm{Gb}}$) is used. This is called a fully connected topology (fully informed PSO) (Mendes et al. 2004), as all particles share information with each other about the best performer of the swarm and parameter *k* equals the size of the whole swarm.

The term *w* of Eq. (3.43) is the inertia weight, a scaling factor employed to control the exploration abilities of the swarm, which scales the current velocity value affecting the updated velocity vector. The inertia weight was not part of the original PSO algorithm (Kennedy and Eberhart 1995), as it was introduced later by Shi and Eberhart (1998) in a successful attempt to improve convergence. Large inertia weights will force larger veloci-

ty updates allowing the algorithm to explore the design space globally. Similarly, small inertia values will force the velocity updates to concentrate in the nearby regions of the design space. The inertia weight can also be updated during iterations, as will be described in detail in Section 3.11.5.

Particles' velocities in each dimension i (i=1, ...,n) are restricted to a maximum velocity v^{max}_i. The vector $\boldsymbol{v}^{max}$ of dimension n holds the maximum absolute velocities for each dimension. It is more appropriate to use a vector rather than a scalar, as in the general case different velocity restrictions can be applied for different dimensions of the particle. If for a given particle j the sum of accelerations of Eq. (3.43) causes the absolute velocity for dimension i to exceed v^{max}_i, then the velocity on that dimension is limited to $\pm v^{max}_i$. The vector parameter $\boldsymbol{v}^{max}$ is employed to protect the cohesion of the system, in the process of amplification of the positive feedback.

Design variables bounds handling

Constraints also apply in the available space for every design variable x_i, as in vector terms $\boldsymbol{x}^L \le \boldsymbol{x} \le \boldsymbol{x}^U$. If, after the velocity update rule of Eq. (3.43), the position update of Eq. (3.44) forces a particle to move outside the bounds for a dimension i ($x_i \le x^L_i$ or $x_i \ge x^U_i$), then the design variable x_i is reset to the closest bound ($x_i = x^L_i$ or $x_i = x^U_i$). In order to avoid considering any points outside the specified design space, the corresponding coefficient v_i of the velocity vector $\boldsymbol{v}$ is reset to zero, to be used for the next iteration.

Main PSO parameters

The basic PSO has only a few parameters to adjust. Table 3.1 contains a list of the main parameters, their typical values and other details.

Table 3.1 Main PSO parameters.

Symbol	Description	Details
NP	Number of particles	A typical range is 10 – 40. For most problems 10 particles is sufficient enough to get acceptable results. For some difficult or special problems the number can be increased to 50-100.
n	Dimension of particles	It is determined by the problem to be optimized.
w	Inertia weight	Usually is set to a value less than 1.0, i.e. 0.95. It can also be updated during iterations.
$\boldsymbol{x}^{\mathrm{L}}$, $\boldsymbol{x}^{\mathrm{U}}$	Vectors containing the lower and upper bounds of the n design variables, respectively	They are determined by the problem to be optimized. Different ranges for different dimensions of particles can be applied in general.
$\boldsymbol{v}^{\max}$	Vector containing the maximum allowable velocity for each dimension during one iteration	Usually is set half the length of the allowable interval for the given dimension: $v^{\max}{}_i = (x^{\mathrm{U}}{}_i - x^{\mathrm{L}}{}_i)/2$. Different values for different dimensions of particles can be applied in general.
c_1, c_2	Cognitive and social parameters	Usually $c_1=c_2=2$. Other values can also be used, provided that $0 < c_1+c_2 < 4$ (Perez and Behdinan 2007a).

Convergence criteria

Due to the repeated process of the PSO search, convergence criteria have to be applied for the termination of the optimization procedure. Two widely adopted convergence criteria are the maximum number of iterations of the PSO algorithm and the minimum error requirement on the calculation of the optimum value of the objective function. The selection of the maximum number of iterations depends generally on the complexity of the optimization problem at hand. The second criterion presumes prior knowledge of the global optimum value, which is feasible for testing or fine-tuning the algorithm in mathematical problems when the optimum is known a priori, but this is certainly not the case in practical structural optimization problems where the optimum is not known a priori.

In the present thesis, together with the maximum number of iterations, we have implemented the convergence criterion connected to the rate of improvement of the value of the objective function for a given number of iterations. If the relative improvement of the objective function over the last k_{f} iterations (including the current iteration) is less or equal to a threshold value f_{m}, convergence is supposed to have been achieved. In mathematical terms, denoting as $Gbest_t$ the best value of the objective function found by the PSO at iteration t, the relative improvement of the objective function can be written for the current iteration t as follows:

$$\frac{Gbest_{t-k_{\mathrm{f}}+1} - Gbest_t}{Gbest_{t-k_{\mathrm{f}}+1}} \leq f_{\mathrm{m}} \tag{3.45}$$

Table 3.2 contains a list of the convergence parameters of the PSO used in this study, with description and details.

Table 3.2 PSO convergence parameters.

Symbol	Description	Details
t_{max}	Maximum number of iterations for the termination criterion.	Determined by the complexity of the problem to be optimized, in conjunction with other PSO parameters (n, NP).
k_f	Number of iterations for which the relative improvement of the objective function satisfies the convergence check.	If the relative improvement of the objective function over the last k_f iterations (including the current iteration) is less or equal to f_m, convergence has been achieved.
f_m	Minimum relative improvement of the value of the objective function.	

Pseudo code of the main PSO algorithm for unconstrained optimization

A description of the pseudo code of the main unconstrained PSO procedure is given in Figure 3.10.

For each particle j

  Initialize particle position by distributing particles randomly in the design space

End

Do

  For each particle j

    Calculate fitness value for current position

    If the current fitness value is better than the best fitness value (*Pbest*) in the particle's history then set current fitness value as the new *Pbest* and current position as the new $\boldsymbol{x}^{\mathrm{Pb},j}$

  End

  Set *Gbest* as the best fitness value of all the particles' *Pbest* and corresponding position as the new $\boldsymbol{x}^{\mathrm{Gb}}$

  For each particle j

    Calculate particle velocity from Eq. (3.43)

    Update particle position from Eq. (3.44)

    If, for any dimension i, $x_i \le x^{\mathrm{L}}{}_i$ or $x_i \ge x^{\mathrm{U}}{}_i$, then set $x_i = x^{\mathrm{L}}{}_i$ or $x_i = x^{\mathrm{U}}{}_i$ respectively and set corresponding $v_i = 0$

  End

While maximum iterations is not attained and the relative improvement of the objective function is greater than f_m over the last k_f iterations

Report results

Figure 3.10 Pseudo-code for the main PSO for unconstrained optimization.

3.11.4 Constraint handling techniques

Although the PSO has been applied for the solution of a number of problems recently, its applications are mainly focused on unconstrained optimization. Very few studies have extended the application of PSO to constrained optimization problems (Hu and Eberhart 2002; Parsopoulos and Vrahatis 2002). A simple approach for PSO would be to recalculate the velocity vector for an infeasible particle using new random numbers r_1 and r_2, until the new position of the particle becomes feasible. This simplistic approach guarantees the feasibility of the final optimum design, yet it has a strong disadvantage as it needs too many calculations of the constraint functions and subsequently of finite element analyses, especially in cases where the feasible region is small compared to the entire design space, making it impractical for structural engineering applications.

Another approach is to avoid taking into account the infeasible designs in the calculation of *Pbest* or *Gbest* for a particle, given that the swarm is initialized in the feasible region (Hu et al. 2003). This "death penalty" approach that guarantees the feasibility of the final optimum has the disadvantage that is does not take into account the degree of the violation of the constraints. Moreover, a search over the feasible region only is usually less efficient than a search over the entire region, as the latter makes it possible to approach the optimum from any direction (Michalewicz 1995).

Venter and Sobieszczanski-Sobieski (2004) proposed a constraint handling mechanism for PSO that redirects the violated designs back to the feasible region. After a particle j has moved to an infeasible position at iteration t, the method modifies the velocity vector $\boldsymbol{v}^j(t)$, by re-setting it to zero. Then, the velocity vector $\boldsymbol{v}^j(t+1)$ for next iteration $t+1$ is obtained from Eq. (3.43) omitting the inertia coefficient $w \cdot \boldsymbol{v}^j(t)$ that equals zero. The new velocity of particle j at iteration $t+1$ is thus only influenced by the best point found so far by the particle ($\boldsymbol{x}^{\mathrm{Pb},j}$) and the current best point found by the entire swarm ($\boldsymbol{x}^{\mathrm{Gb}}$). Given that both these best points are feasible, the new velocity vector will point back to a feasible region of the design space, ensuring in most cases that the particle is directed back to the feasible space, or at least closer to the feasibility boundary. This method is also simple, but has the disadvantage that it does not guarantee feasibility of the particles and as a result, there is no guarantee that at the computed optimum solution all constraints will be satisfied.

Some researchers attempted to solve the constrained problem indirectly by transforming it to an unconstrained problem using the traditional penalty function strategy (Parsopoulos and Vrahatis 2002; Perez and Behdinan 2007). The penalty function is an effective auxiliary tool to deal with constrained problems in general and has been a popular approach because of its simplicity and ease of implementation. Yeniay (2005) examined various penalty function methods for GAs, highlighting the strengths and weaknesses of each method.

In the present thesis, we propose a simple yet effective multiple linear segment penalty function to deal with constraints. Consider a structural optimization problem where dis-

placement and stress constraints are imposed. For a given design $\boldsymbol{x}$, the corresponding objective function value is computed and a finite element analysis is performed for the constraints check where each structural element is checked for stress violation, and each model node is checked for displacement violation. If no violation is detected, then no penalty is imposed on the objective function $f(\boldsymbol{x})$. If any of the constraints are violated, a penalty is applied to the objective function and the value of the penalty is related to the degree of violation of the constraints.

For the typical constraint k in structural optimization of the form

$$g_k(\boldsymbol{x}) = |q_k(\boldsymbol{x})| - q_{\text{allow},k} \le 0 \tag{3.46}$$

where $q_k(\boldsymbol{x})$ is a response measure (usually stress or displacement) for design vector $\boldsymbol{x}$ and $q_{\text{allow},k}$ its maximum allowable absolute value, the penalty function $\Phi_k(\boldsymbol{x})$ for this constraint is defined as:

$$\Phi_k(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad \dfrac{|q_k(\boldsymbol{x})|}{q_{\text{allow},k}} \le 1 \\ \dfrac{|q_k(\boldsymbol{x})|}{q_{\text{allow},k}} & \text{if} \quad \dfrac{|q_k(\boldsymbol{x})|}{q_{\text{allow},k}} > 1 \end{cases} \tag{3.47}$$

It should be noted that $q_k(\boldsymbol{x})$ is taken as the maximum (worst) value of the corresponding response measure among all nodes or elements of the model. For example, if the k-th constraint is a stress constraint of the type $|\sigma| \le \sigma_{\text{allow}}$ that applies for all N_e model elements, then for this constraint a single response measure is calculated as:

$$q_k(\boldsymbol{x}) = \max_{i=1}^{N_e} \{|\sigma_i|\} \tag{3.48}$$

Figure 3.11 gives a graphical representation of the above formula.

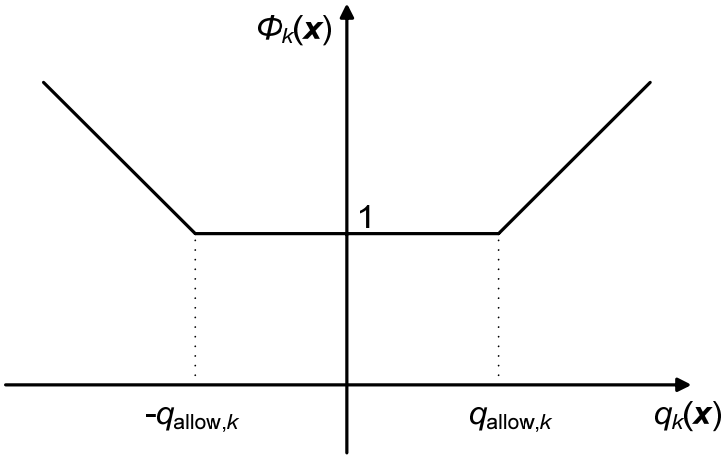


Figure 3.11 A multiple linear segment penalty function.

Having obtained the penalty function factors for all violated constraints, the penalized fitness value of a design $\boldsymbol{x}$ is obtained by multiplying the objective function (structural weight or structural material volume) to be minimized by the maximum penalty factor among all m constraints:

$$f_{\mathrm{p}}(\boldsymbol{x}) = f(\boldsymbol{x}) \cdot \max\{\Phi_k(\boldsymbol{x})\}, \quad k = 1, \ldots, m \tag{3.49}$$

where f_{p} is the new fitness (penalized objective function). The resulting penalized objective function quantitatively represents the extent of the violation of constraints and provides a relatively meaningful measurement of the performance of each solution.

Using the above formulation, there is a case where the penalized objective function $f_{\mathrm{p}}(\boldsymbol{x})$ can obtain a better value compared to the global optimum $Gbest_t$ found by the entire swarm until iteration t. This will result in resetting $Gbest_t$ to an infeasible design. Indeed, this can happen for an infeasible design if $\max\{\Phi_k(\boldsymbol{x})\} < Gbest_t / f(\boldsymbol{x})$. In order to avoid this undesirable case, $Gbest_t$ is used instead of $f(\boldsymbol{x})$ in Eq. (3.49), when $f(\boldsymbol{x}) < Gbest_t$ and $\max\{\Phi_k(\boldsymbol{x})\} > 1$ (infeasible design). In this sense, $Gbest_t$ is penalized instead of $f(\boldsymbol{x})$, for infeasible designs with objective functions $f(\boldsymbol{x})$ better than $Gbest_t$. This ensures that the best design found by the swarm will always stay in the feasible region, as will be shown in the numerical applications section.

3.11.5 PSO for constrained structural optimization

Two important features which require special attention when dealing with practical engineering optimization problems are the improvement of the convergence rate and the handling of the problem constraints. As described below, different modifications can be made to the original algorithm to address these features making it much capable of dealing with more demanding constrained optimization problems such as those encountered frequently in optimum design of engineering structures.

Inertia weight update

The PSO global convergence is affected by the degree of local/global exploration provided by the c_1 and c_2 parameters, while the relative rate of convergence is affected by the inertia weight parameter. Studies have shown that for a fixed inertia value there is a significant reduction in the algorithm convergence rate as iterations progress. This is the consequence of excessive momentum in the particles, which results in large step sizes that overshoot the best design areas. During the initial optimization stages, a large inertia weight is needed in order for the design space to be searched thoroughly. Once the most promising areas of the design space have been discovered and the convergence rate starts to slow down, the inertia weight should be reduced, in order for the particles' momentum to decrease allowing them to concentrate in the best design areas. In order to accomplish the above strategy, Shi and Eberhart (1998) proposed a time-dependent value of the inertia weight. A commonly used inertia update rule is the linearly-decreasing, calculated by the formula:

$$w_{t+1} = w_{\max} - \frac{w_{\max} - w_{\min}}{t_{\max}} \cdot t \tag{3.50}$$

where t is the iteration number (starting from iteration 0), w_{max} and w_{min} are the maximum and minimum values, respectively, of the inertia weight. In general, the linearly decreasing inertia weight has shown better performance than the fixed one.

In the present thesis, a new non-linear weight update strategy is adopted: the total allowed iterations $t_{\max}$ are divided into three stages. At the end of each stage, the change (reduction) of w compared to the one at the end of the previous stage has to be a_w times its value. Given that, we can define the value of w at $t_{\max}/3$ and $2 \cdot t_{\max}/3$ iterations. A cubic polynomial is then calculated that interpolates the four points (starting point $(0, w_{\max})$, ending point $(t_{\max}, w_{\min})$ and two intermediate points, $(1/3 \cdot t_{\max}, w_{\max}-a_w^2 \cdot b)$ and $(2/3 \cdot t_{\max}, w_{\min}+b)$), where b is the reduction of w for the third stage, as shown in Figure 3.12.

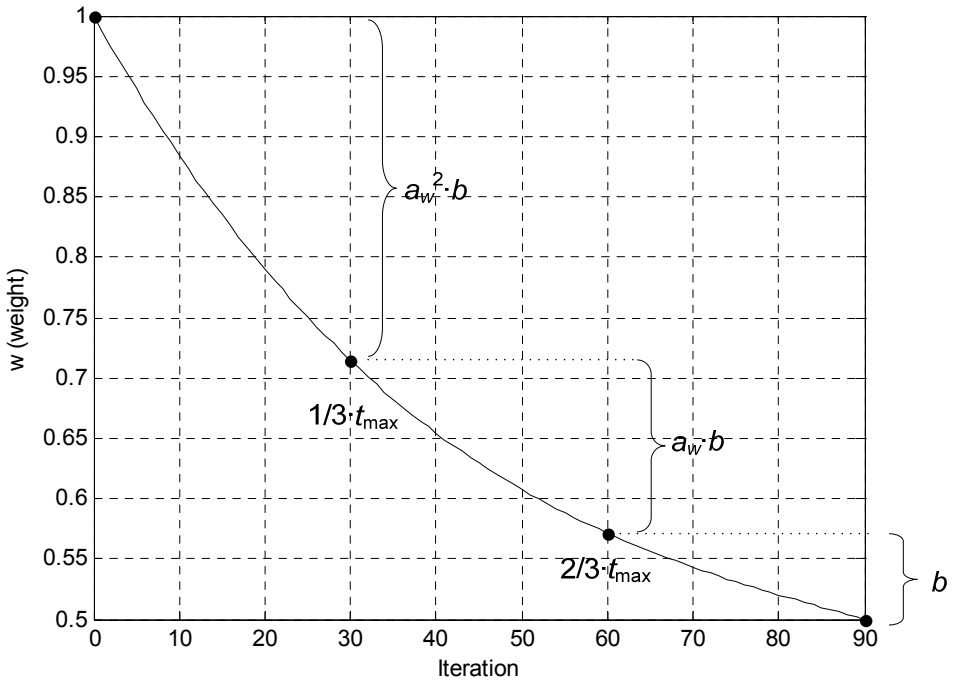


Figure 3.12 The proposed non-linear weight update rule drawn for $t_{\max}=90$, $w_{\min}=0.5$, $w_{\max}=1$ and $a_w=2$.

The quantity b is not a new parameter as it is dependent on $w_{\max}$, $w_{\min}$ and a_w and can be easily calculated as

$$b + a_w \cdot b + a_w^2 \cdot b = w_{\max} - w_{\min} \Leftrightarrow \tag{3.51}$$

$$b = \frac{w_{\max} - w_{\min}}{a_w^2 + a_w + 1} \tag{3.52}$$

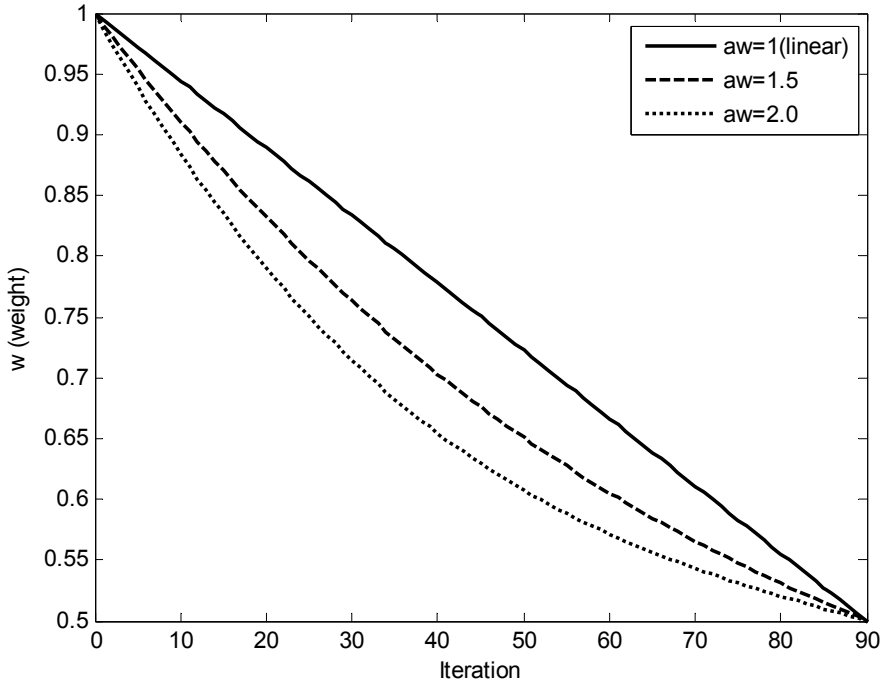


Figure 3.13 The proposed non-linear weight update rule for different a_w (1.0, 1.5, 2.0).

Compared to the linear update rule, the proposed non-linear 3rd order formulation has the advantage of a fast reduction of the inertia weight in the first stage of the optimization, while in the vicinity of the optimum, the reduction becomes slower, as shown in Figure 3.13. This type of behavior is in most cases favorable in PSO optimization, as will be shown in the numerical applications section. The linear update rule can be obtained as a special case by setting a_w=1. Typical values for a_w are in the interval [1.0,2.0]. Values smaller than 1 should not be considered as they would lead to the opposite undesirable result: a small reduction of the inertia weight in the first stages and a fast reduction near the optimum.

Pseudo code of the proposed PSO algorithm for constrained structural optimization

The proposed PSO pseudo code for constrained structural optimization is described in Figure 3.14.

For each particle j

Initialize particle position by distributing particles randomly in the design space

End

Do

For each particle j

Calculate all response measures $q_k(\boldsymbol{x})$ by performing a finite element analysis and computing the corresponding $\Phi_k(\boldsymbol{x})$ from Eq. (3.47) for every constraint k

Calculate fitness value f_p from Eq. (3.49)

If the current fitness value f_p is better than the best fitness value (*Pbest*) in the particle's history then set current fitness value f_p as the new *Pbest* and current position as the new $\boldsymbol{x}^{Pb,j}$

End

Set *Gbest* as the best penalized fitness value of all the particles' *Pbest* and corresponding position as the new $\boldsymbol{x}^{Gb}$

For each particle j

Calculate particle velocity from Eq. (3.43) using the non-linear cubic weight update rule

Update particle position from Eq. (3.44)

If, for any dimension i, $x_i \le x^L{}_i$ or $x_i \ge x^U{}_i$, then set $x_i = x^L{}_i$ or $x_i = x^U{}_i$ respectively and set corresponding $v_i = 0$

End

While maximum iterations is not attained and the relative improvement of the objective function is greater than f_m over the last k_f iterations

Report results

Figure 3.14 The proposed PSO pseudo-code for constrained structural optimization.

3.11.6 PSO related to mathematical methods

By substituting $\boldsymbol{x}^j(t)$ from Eq. (3.44) to Eq. (3.43) we obtain:

$$\boldsymbol{x}^j(t+1) = \boldsymbol{x}^j(t) + w\boldsymbol{v}^j(t) + c_1\boldsymbol{r}_1 \circ \left(\boldsymbol{x}^{Pb,j} - \boldsymbol{x}^j(t)\right) + c_2\boldsymbol{r}_2 \circ \left(\boldsymbol{x}^{Gb} - \boldsymbol{x}^j(t)\right) \tag{3.53}$$

The above equation can be rewritten as:

$$\boldsymbol{x}^j(t+1) = \boldsymbol{x}^j(t) + w\boldsymbol{v}^j(t) + (c_1\boldsymbol{r}_1 + c_2\boldsymbol{r}_2) \circ \left(\frac{c_1\boldsymbol{r}_1 \circ \boldsymbol{x}^{Pb,j} + c_2\boldsymbol{r}_2 \circ \boldsymbol{x}^{Gb}}{c_1\boldsymbol{r}_1 + c_2\boldsymbol{r}_2} - \boldsymbol{x}^j(t)\right) \tag{3.54}$$

The above equation has the same structure as the gradient line-search used in convex unconstrained optimization:

$$\boldsymbol{x}^j(t+1) = \tilde{\boldsymbol{x}}^j(t) + \boldsymbol{a} \circ \boldsymbol{p}_k \tag{3.55}$$

where

$$\tilde{\boldsymbol{x}}^j(t) = \boldsymbol{x}^j(t) + w\boldsymbol{v}^j(t) \tag{3.56}$$

$$\boldsymbol{a} = c_1\boldsymbol{r}_1 + c_2\boldsymbol{r}_2 \tag{3.57}$$

$$\boldsymbol{p}_k = \frac{c_1\boldsymbol{r}_1 \circ \boldsymbol{x}^{\mathrm{Pb},j} + c_2\boldsymbol{r}_2 \circ \boldsymbol{x}^{\mathrm{Gb}}}{c_1\boldsymbol{r}_1 + c_2\boldsymbol{r}_2} - \boldsymbol{x}^j(t) \tag{3.58}$$

So the process can be viewed as a traditional line-search procedure dependent on a stochastic step size $\boldsymbol{\alpha}$ and a stochastic search direction $\boldsymbol{p}_k$ (Perez and Behdinan 2007a), where:

i. The stochastic step size $\boldsymbol{\alpha}$ is limited only by the selection of the social and cognitive parameters c_1 and c_2 and knowing that $\boldsymbol{r}_1, \boldsymbol{r}_2 \in [0,1]$ it will belong to the interval $[0, c_1+c_2]$ with a mean value of $(c_1+c_2)/2$.

ii. The stochastic search direction $\boldsymbol{p}_k$ belongs to the interval $\left[-\boldsymbol{x}^j(t), \frac{c_1\boldsymbol{x}^{\mathrm{Pb},j} + c_2\boldsymbol{x}^{\mathrm{Gb}}}{c_1 + c_2} - \boldsymbol{x}^j(t) \right]$, which is dependent not only on the social and cognitive parameters c_1 and c_2, but also in the best design space location the particle has seen ($\boldsymbol{x}^{\mathrm{Pb},j}$) and the best design space location all the swarm has seen ($\boldsymbol{x}^{\mathrm{Gb}}$).

3.12 Hybrid optimization algorithms

Various hybrid optimization algorithms that combine evolutionary computation techniques with deterministic procedures for numerical optimization problems have been proposed in the past. Papadrakakis et al. (1999) proposed a hybrid ES-SQP scheme for structural shape optimization with very satisfactory results. Waagen et al. (1992) proposed a combination of evolutionary programming and the direction set method of Hooke and Jeeves (1961) applied to unconstrained mathematical test functions. Myung et al. (1995) considered a similar to Waagen et al. (1992) approach, but they experimented with constrained mathematical test functions. They combined a floating-point evolutionary programming technique, with a method-developed by Maa and Shanblatt (1992) applied to the best solution found by the evolutionary programming technique. Papadrakakis and Lagaros (2000) implemented a hybrid GA-MP scheme for structural sizing optimization.

Hybrid PSO methods have been also proposed recently. Kaveh and Talatahari (2008) implemented a hybrid PSO and ant colony optimization for the design of truss structures. Dimopoulos (2007) proposed a hybrid GA-PSO scheme for optimizing mathematical functions and optimally designing a welded beam and a pressure vessel for minimum cost, while Zhang et al. (2007) proposed a hybrid PSO - back-propagation algorithm for feed-forward neural network training. The PSO has been also combined with mathematical methods in various ways. Izui et al. (2005; 2007) combined a PSO scheme with gra-

dients, where the members of the swarm were divided into Sequential Linear Programming (SLP) and PSO individuals. Chen et al. (2007) combined the PSO algorithm with a conjugate gradient based local search method for the identification of nonlinear systems. Coelho and Mariani (2007) combined the PSO with a Quasi-Newton local search method for solving an economic dispatch problem. Das et al. (2006) proposed adding a local search component to PSO to improve its convergence speed for estimating the parameters of a gene network model. Victoire and Jeyakumar (2004) combined the PSO with SQP for solving the economic dispatch problem, where SQP was used to fine tune the solution obtained by the PSO.

3.12.1 Hybrid PSO-SQP methodology

The numerical tests performed with the PSO algorithm have shown rapid convergence of the method during the initial stages of the global search, but at the neighborhood of the global optimum, the search process becomes rather slow, a typical behavior of all evolutionary type optimization algorithms. On the contrary, studies have shown that gradient based methods can achieve faster convergence speed around global optimum and, at the same time, the convergence accuracy can be higher.

An important feature of the SQP optimizer is that it captures relatively fast the right path to the nearest optimum. Yet, unless a good model initialization is provided, the algorithm can converge to a local suboptimum. Therefore, global algorithms - less vulnerable to local optima attractors and therefore more reliable in obtaining the global optimum for non-convex optimization problems, are frequently proposed, but have often exhibited inacceptable slow convergence rates due to their random search, especially near the area of the global optimum.

In an effort to increase the robustness and the computational efficiency of the optimization procedure, hybrid algorithms can benefit from the advantages of both methodologies and alleviate their particular drawbacks, as was described in detail in Section 3.12. In the present thesis, a novel hybrid algorithm combining the PSO algorithm with a gradient-based quasi-Newton Sequential Quadratic Programming algorithm, is proposed for the optimization of engineering structures. The hybrid optimization algorithm can make use of not only the strong global searching ability of the PSO, but also the strong local searching ability of the SQP algorithm. It is divided into two separate phases:

i. During the first phase, the PSO explores the design space thoroughly and detects the neighborhood of the global optimum;

ii. When the PSO process terminates, the second phase starts by applying the SQP algorithm starting from the best estimate of the PSO and using gradient information to accelerate convergence towards the global optimum.

In the first phase, a relaxed termination criterion should be used for PSO, as there is no need for PSO to find the exact global optimum, but rather to find its neighborhood in

the design space. The exact value of the global optimum will be found in the second phase, by the SQP optimizer. The combined algorithm is referred to as PSO-SQP.

Enhancements are also proposed for the PSO algorithm for constrained structural optimization itself, with the implementation of the non-linear cubic weight update rule, described in Section 3.11.4, and the simple yet effective linear segment penalty function constraint handling technique, described in Section 3.11.5. The numerical results show that the proposed hybrid PSO–SQP algorithm performs better than the standard PSO algorithm as well as other established EA algorithms in terms of convergence speed and quality of final results achieved. The method is applied to structural engineering optimization problems where the aim is to find the optimum design of a structure under specific loads. The structures considered are plane or space trusses, the objective function is the weight of the structure, while the constraints refer to restrictions in the maximum values of stresses and displacements. The constraints are checked by a finite element analysis for every candidate optimum design.

Chapter 4

4 Multi-objective Optimization

Although many optimization studies deal with only one objective function, this approach is often not realistic for practical engineering applications. Many real-world problems involve several incommensurable and often competing objectives to be handled simultaneously. In this chapter, the principles of multi-objective (or vector) optimization are outlined, the basic concepts are defined and the methodologies for handling *Multi-objective Optimization Problems* (MOPs) are demonstrated.

4.1 The concept of multi-objective optimization

Consider an optimization problem with multiple objective functions to be minimized. If a solution can be found that minimizes all the objective functions simultaneously, then actually a *Single-objective Optimization Problem* (SOP) is at hand, as the optimal solution for any objective is also the optimum for any other objective. In SOPs, the optimal solution is usually clearly defined, while this does not hold in MOPs, where in the usual case individual optima corresponding to the various objective functions are sufficiently different to each other. In that case, the objective functions are conflicting with each other and cannot be optimized simultaneously. Instead, a satisfactory trade-off has to be found, leading to a set of optimal solutions rather than a single solution for the optimization problem. The optimality of these solutions can be understood in a wider sense, in that in the whole search space there are no other solutions superior to them when all the objectives are considered simultaneously.

4.2 Formulation of a multi-objective optimization problem

The formulation of a general multi-objective optimization problem of m objectives can be written as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{f}(\boldsymbol{x}), \quad \boldsymbol{x} = [x_1, \ldots, x_n]^{\mathrm{T}}, \quad \boldsymbol{f} \in \mathbb{R}^m$$
$$\text{Subject to}$$
$$\boldsymbol{g}(\boldsymbol{x}) \le 0, \quad \boldsymbol{g} \in \mathbb{R}^p \tag{4.1}$$
$$\boldsymbol{h}(\boldsymbol{x}) = 0, \quad \boldsymbol{h} \in \mathbb{R}^q$$
$$x_i \in X_i \quad \text{for } i = 1, \ldots, n$$

where:

- $\boldsymbol{x} = [x_1, \ldots, x_n]^{\mathrm{T}}$ is the vector of the n design variables.
- X_i is the set of x_i, which may be continuous, discrete or integer. The whole design space for the n design variables can be denoted as $\mathcal{X}$.
- $\boldsymbol{f}(\boldsymbol{x})^{\mathrm{T}} = [f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]$ is the vector function of m objectives to be minimized.
- $\boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} = [g_1(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x})]$ is the vector function of p inequality constraints.
- $\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}} = [h_1(\boldsymbol{x}), \ldots, h_q(\boldsymbol{x})]$ is the vector function of q equality constraints.

4.3 Definitions for MOPs

Feasible set: The *feasible set* $\mathcal{F}$ is defined as the set of design vectors $\boldsymbol{x}$ that satisfy the constraints of the optimization problem of Eq. (4.1). It is obvious that $\mathcal{F}$ is a subset of $\mathcal{X}$.

$$\mathcal{F} = \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{g}(\boldsymbol{x}) \le 0 \mid \boldsymbol{h}(\boldsymbol{x}) = 0\} \tag{4.2}$$

$$\mathcal{F} \subseteq \mathcal{X} \tag{4.3}$$

The image of $\mathcal{F}$, i.e. the feasible region in the objective space, is called the *criterion space*, denoted as $\mathcal{Y}_{\mathcal{F}} = \boldsymbol{f}(\mathcal{F})$. All equality constraints, if they exist (regardless of the value of $\boldsymbol{x}$ used), are considered active at all points of the feasible set $\mathcal{F}$.

Feasible design: A design vector $\boldsymbol{x}$ is feasible if and only if it belongs to the feasible set $\mathcal{F}$.

In single-objective optimization, the feasible set $\mathcal{F}$ is completely ordered according to the single value of the objective function f. For two solutions $\boldsymbol{x}^{\mathrm{a}}, \boldsymbol{x}^{\mathrm{b}} \in \mathcal{F}$, either $f(\boldsymbol{x}^{\mathrm{a}}) \le f(\boldsymbol{x}^{\mathrm{b}})$ or $f(\boldsymbol{x}^{\mathrm{b}}) \le f(\boldsymbol{x}^{\mathrm{a}})$ (or both) and the goal is to find the solution that gives the minimum value of the objective function f. However, when several objectives are considered, the feasible set $\mathcal{F}$ is in the general case not completely, but partially ordered (Pareto 1896), and two solutions cannot be classified in a univocal manner. This is illustrated in Figure 4.1 depicting the objective space for an unconstrained MOP with two objective functions f_1 and f_2.

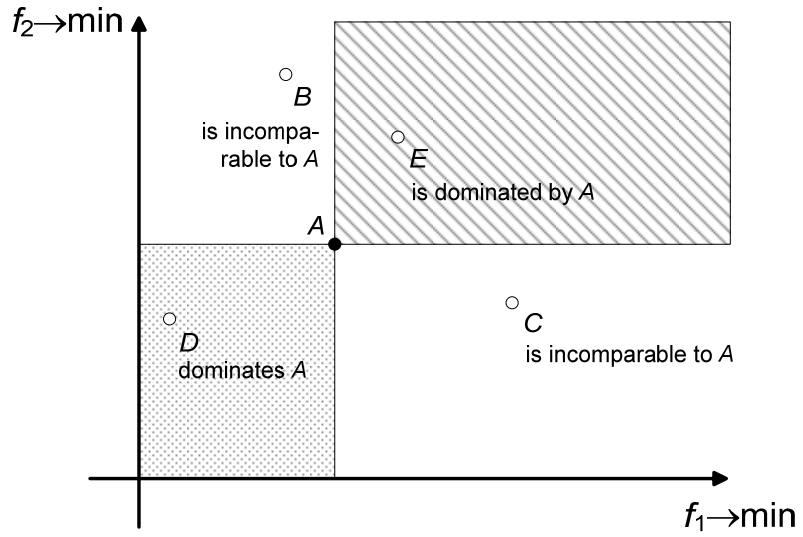


Figure 4.1 Dominated, dominating and incomparable regions with respect to point *A* in the objective space.

The solution $\boldsymbol{x}^A$ represented by point *A* (reference point) in the objective space is clearly better than the solution $\boldsymbol{x}^E$ represented by point *E* as it gives better values for both objective functions f_1 and f_2, as in mathematical terms, $f_1(\boldsymbol{x}^A)<f_1(\boldsymbol{x}^E)$ and $f_2(\boldsymbol{x}^A)<f_2(\boldsymbol{x}^E)$. On the other hand, the solution $\boldsymbol{x}^D$ represented by point *D* is superior to the one represented by *A*, as $f_1(\boldsymbol{x}^D)<f_1(\boldsymbol{x}^A)$ and $f_2(\boldsymbol{x}^D)<f_2(\boldsymbol{x}^A)$, while the solutions represented by *B* and *C* are incomparable to *A* as each one is better than *A* for one of the objective functions, f_1 and f_2 respectively, $f_1(\boldsymbol{x}^B)<f_1(\boldsymbol{x}^A)$ and $f_2(\boldsymbol{x}^A)<f_2(\boldsymbol{x}^B)$ while $f_1(\boldsymbol{x}^A)<f_1(\boldsymbol{x}^C)$ and $f_2(\boldsymbol{x}^C)<f_2(\boldsymbol{x}^A)$.

In order to express this situation mathematically, the equality and inequality notions for a SOP are extended to objective vectors using the following definitions:

Weak Pareto dominance: An objective vector $\boldsymbol{u}$ is said to *weakly dominate* objective vector $\boldsymbol{v}$ ($\boldsymbol{u} \succeq \boldsymbol{v}$)[1] if and only if

$$u_i \le v_i \quad \forall \; i = 1, \ldots, n \tag{4.4}$$

Pareto dominance: An objective vector $\boldsymbol{u}$ is said to *dominate* objective vector $\boldsymbol{v}$ ($\boldsymbol{u} \succ \boldsymbol{v}$) if and only if it weakly dominates objective vector $\boldsymbol{v}$ (Eq.(4.4)) and $u_i<v_i$ for at least one $i=1,\ldots,n$.

Strict Pareto dominance: An objective vector $\boldsymbol{u}$ is said to *strictly dominate* objective vector $\boldsymbol{v}$ ($\boldsymbol{u} \succ\succ \boldsymbol{v}$) if and only if $u_i < v_i \;\; \forall \;\; i = 1, \ldots, n$.

[1] Note that these definitions for the notation $\succeq$, $\succ$ and $\succ\succ$ are given for a *minimization* problem.

Incomparability: Two obj5ective vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are said to be *incomparable* to each other ($\boldsymbol{u} \prec\succ \boldsymbol{v}$) if neither $\boldsymbol{u} \succeq \boldsymbol{v}$ nor $\boldsymbol{v} \succeq \boldsymbol{u}$. Some authors also use the term *indifferent* to describe the same notion.

In Figure 4.1 the upper right rectangle, where point *E* belongs to, encapsulates the region in the objective space that is dominated by the objective vector represented by reference point *A*. The lower left rectangle, where point *D* belongs to, contains the objective vectors that dominate the solution associated with *A*. The other two rectangles, the upper left and the lower right, where points *B* and *C* belong to, respectively, contain solutions that are incomparable to *A*. Based on the concept of Pareto dominance, the definitions of both local Pareto optimality (Wan 1975) and global Pareto optimality for MOPs can be introduced as follows:

Local Pareto optimality: A design vector $\boldsymbol{x}^{\mathrm{L}} \in \mathcal{F}$ is said to be a *Local Pareto optimal design vector* if and only if there exists a neighborhood $\aleph$ of $\boldsymbol{x}^{\mathrm{L}}$ in which there exists no other $\boldsymbol{x} \in \aleph$ such that $\boldsymbol{f}(\boldsymbol{x}) \succ \boldsymbol{f}(\boldsymbol{x}^{\mathrm{L}})$.

(Global) Pareto optimality: A design vector $\boldsymbol{x}^{*} \in \mathcal{F}$ is a *Pareto optimal design vector* if and only if there exists no other $\boldsymbol{x} \in \mathcal{F}$ such that $\boldsymbol{f}(\boldsymbol{x}) \succ \boldsymbol{f}(\boldsymbol{x}^{*})$

In other words, a design vector $\boldsymbol{x}^{*} \in \mathcal{F}$ is a Pareto optimal design vector if and only if there exists no other $\boldsymbol{x} \in \mathcal{F}$ such that

$$f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^{*}) \ \ \forall \ \ i = 1,\ldots,n \tag{4.5}$$

with $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^{*})$ for at least one objective *i*.

Extending the above definition, the image of a Pareto optimal design vector in the objective space can be called a *Pareto optimal objective vector*. From the above definitions it is clear that a global Pareto optimal design vector is necessarily also a local Pareto optimal design vector, while the converse does not hold in general. The converse is true only in the special case where all the functions involved are convex functions.

Pareto set: *Pareto set* $\mathcal{P}^{*}$ is the set of all the Pareto optimal design vectors $\boldsymbol{x}^{*} \in \mathcal{F}$.

In other words, Pareto set $\mathcal{P}^{*}$ is the set of all the design vectors that correspond to non-dominated objective vectors (solutions):

$$\mathcal{P}^{*} = \{\boldsymbol{x}^{*} \in \mathcal{F} \mid \text{There is no } \boldsymbol{x} \in \mathcal{F} \text{ such that } \boldsymbol{f}(\boldsymbol{x}) \succ \boldsymbol{f}(\boldsymbol{x}^{*})\} \tag{4.6}$$

Pareto front: *Pareto front* (PF) is the image of the Pareto set $\mathcal{P}^{*}$ in the objective function space. In other words, Pareto front is the set of all Pareto optimal objective vectors.

Figure 4.2 depicts the feasible region in the objective space for a minimization problem with two objective functions f_1 and f_2. The lower left bold line of the feasible region boundary is the Pareto front. White points represent objective vectors that correspond to feasible design vectors, but are dominated by at least one other objective vector. On the other hand, the black points along the front represent some of the Pareto optimal objective vectors, which are not dominated by any other objective vector in the feasible re-

gion. All black points correspond to objective vectors that are incomparable to each other. Thus, a black point is optimal in the sense that it cannot be improved in one objective without causing deterioration in the other objective.

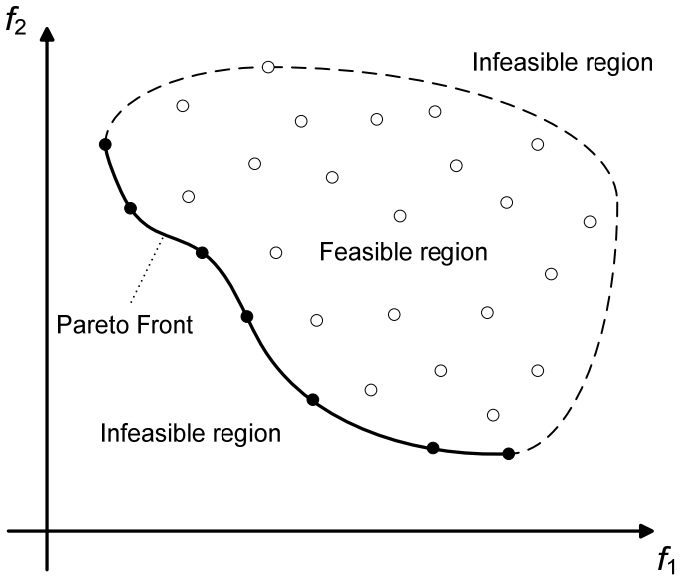


Figure 4.2 Feasible region and corresponding Pareto Front in the objective space for a two-objective minimization problem.

It is clear that in the multi-objective case there is no single optimal solution for the optimization problem, but rather a set of optimal trade-offs that are represented by the Pareto Front. None of these points can be identified as better than another point of the front. They are all candidates for selection depending on the decision maker's preference. It should be noted that each point along the Pareto Front in the objective space has a corresponding point (or maybe more than just one point) in the design space, yet the graphical interpretation of non-dominance applies only in the objective space.

4.4 Conflict and criteria

The engineer looking for the optimum design of a structure is faced with the question of selecting the most suitable criteria for measuring the economy, strength, serviceability or any other factor that affects the performance of the structure. Any quantity that has an influence on the structural performance can be considered as a criterion. One important basic property in the multi-criterion formulation is the conflict that should exist among the various criteria to be taken into consideration. Only those quantities that are conflicting with each other should be treated as independent criteria whereas the others can be combined into a single criterion representing the whole group. Below there are some definitions on criteria conflict.

Local collinearity: Two objective functions f_i and f_j are called *locally collinear* with no conflict at point $\boldsymbol{x}$ if there is $c>0$ such that

$$\nabla f_i(\boldsymbol{x}) = c \cdot \nabla f_j(\boldsymbol{x}) \tag{4.7}$$

Otherwise, the functions are called *locally conflicting* at point $\boldsymbol{x}$. According to this definition any two criteria are locally conflicting at a point of the design space if their maximum improvement is achieved in different directions.

Global conflict: Two objective functions f_i and f_j are called *globally conflicting* in the feasible region $\mathcal{F}$ of the design space if the two single-objective optimization problems

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f_i(\boldsymbol{x}) \quad \boldsymbol{x} = [x_1,\ldots,x_n]^{\mathrm{T}} \tag{4.8}$$

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f_j(\boldsymbol{x}) \quad \boldsymbol{x} = [x_1,\ldots,x_n]^{\mathrm{T}} \tag{4.9}$$

have different optimal solutions in the feasible region $\mathcal{F}$ of the design space.

4.5 Search and decision making

Two conceptually distinct types of problems can be identified in solving a MOP; (i) search; and (ii) decision making (Horn 1997). Search refers to the optimization process in which the optimizer searches the feasible set for Pareto optimal solutions. Decision making addresses the problem of selecting the suitable compromise solution among the Pareto set. This requires a human *Decision Maker* (DM) to make the trade-off by selecting a single solution among the optimal set.

Depending on how search and decision process are combined, multi-objective optimization methods can be classified into three categories (Horn 1997; Hwang and Masud 1979):

i. **Decision making before search**: The objectives of the MOP are aggregated and the multi-objective problem is transformed into a single-objective one. This implicitly includes preference information given by the DM.

ii. **Decision making after search**: No preference information is given. After the search phase has been concluded and the Pareto set has been defined, the DM makes the trade-off and the final choice among the different optimal solutions.

iii. **Decision making during search**: Preferences can be articulated by the DM during the optimization process. After each optimization step, a number of alternative trade-offs is presented and the DM specifies further preference information, guiding the search process.

The standard approach of aggregating the various objectives into a single criterion has the advantage that various well-established single-objective optimization strategies can be applied without any modification. However, it requires a profound domain knowledge

which is usually not available. Performing the search before decision making overcomes this drawback, but excludes preference articulation by the DM which might reduce the search space complexity. Finally, the integration of search and decision making is a promising way to combine the other two approaches, exploiting the advantages of both.

4.6 Methods for solving MOPs

In real-world structural optimization problems, two difficulties are usually encountered; (i) multiple conflicting objectives may exist; and (ii) the search space can be complicated and non-convex. In order to overcome these difficulties, efficient optimization strategies are required. There is a large variety of analytical techniques for solving multi-objective optimization problems, starting from a few decades ago. Cohon (1978) reviewed many of the available methods of that time. Hwang and Masud (1979) illustrated a large number of methods by solving numerical examples in detail. Zeleny (1982) provided a comprehensive treatment of the entire multicriteria endeavor. Stadler (1988) offered broad coverage of the field with many examples from engineering and science. Chankong and Haimes (1983) included a rigorous development of most multicriteria techniques of that time. Steuer (1986) provided an especially useful review of multicriteria linear programming theories and algorithms.

Miettinen (1999) gave a thorough review of non-linear multi-objective optimization theories and methods, while Zitzler (1999) examined various evolutionary algorithms for multi-objective optimization. Various methods have been also proposed lately for treating structural multi-objective optimization problems (Coelho et al. 2003; Coello Coello 2000; Deb et al. 2002; Greiner et al. 2004; Luh and Chueh 2004; Marler and Arora 2004; Zitzler et al. 2000). The large number of multi-objective optimization methods can be classified in many ways according to different criteria. Hwang and Masud (1979), followed by Buchanan (1986), Lieberman (1991), and Miettinen (1999), classified the methods according to the participation of the decision-maker in the solution process: (i) no preference methods; (ii) a priori methods; (iii) interactive methods; and (iv) a posteriori methods. Rosenthal (1985) suggested three classes of solution methods: (i) partial generation of the Pareto set; (ii) explicit value function maximization; and (iii) interactive implicit value function maximization. In Carmichael (1981), methods were classified according to whether (i) a composite single objective function; (ii) a single objective function with constraints; or (iii) many single objective functions were the basis for the approach.

According to Marler and Arora (2004) MOP methods can be classified into the following categories:

i. Methods with a priori articulation of preferences;

ii. Methods with a posteriori articulation of preferences;

iii. Methods with no articulation of preferences.

In the case of the methods with a priori articulation of preferences, which are the most popular methods, a scalarizing function with weight parameters is usually used and a series of optimization runs have to be executed, as the multi-objective optimization problem is modified into many single objective optimization sub-problems. In general, when using scalarizing functions local Pareto optimal solutions are obtained. Global Pareto optimality can be guaranteed only when the objective functions and the feasible region are both convex or quasi-convex and convex, respectively. For non-convex cases, such as the majority of structural multi-objective optimization problems, a global single objective optimizer is required.

The quality of the Pareto front curve obtained from any optimization method can be assessed according to the following criteria in general:

i. Distance from the exact Global Pareto front curve;

ii. Number of Pareto optimum solutions;

iii. Distribution of the Pareto optimum solutions along the front curve.

4.7 Standard methods

Standard methods for generating the Pareto-optimal set combine the various objectives into a single, parameterized objective function by analogy to decision making before search. However, the parameters of this function are not set by the decision maker but systematically varied by the optimizer. Several optimization runs with different parameter settings are performed in order to achieve a set of solutions which approximates the Pareto optimal set. Basically, this procedure is independent of the incorporated optimization algorithm, as any single-objective optimization algorithm can be used for the solution of the sub-problems.

Some representatives of this class of methods are the *Linear Weighting Method* (LWM) (Cohon 1978), the *Distance Function Method* (DFM) (Zeleny 1982), the *Constraint Method* (CM) (Haimes and Hall 1974), the *Goal Programming* (GP) method (Steuer 1986) and the minmax approach (Koski 1984).

Standard methods are attractive and popular because of the fact that they transform the original MOP into a series of SOPs, and as a result well-studied algorithms for SOPs can be implemented. However, there are limitations and difficulties in their use (Zitzler 1999), as:

i. Some techniques may be sensitive to the shape of the Pareto Front (mainly weighting methods);

ii. Problem knowledge may be required which may not be available;

iii. They require several optimization runs to obtain an approximation of the Pareto-optimal set. As these runs are performed independently from each other, synergies

can usually not be exploited which, in turn, may cause high computational overhead (Deb 1999).

4.7.1 Linear Weighting Method (LWM)

The Linear Weighting Method (Cohon 1978), combines all the objectives into a single scalar parameterized objective function by using weighting coefficients. Let w_i be the weighting coefficient for the i-th objective function ($i=1,\ldots,m$). Then the optimization problem of Eq. (4.1) can be written as

$$\min_{\boldsymbol{x}\in\mathcal{F}} \sum_{i=1}^{m} w_i f_i(\boldsymbol{x}) \tag{4.10}$$

With no loss of generality the following normalization of the weighting coefficients is employed

$$\sum_{i=1}^{m} w_i = 1 \tag{4.11}$$

If for some given values of the weights w_i ($i=1,\ldots,m$) the condition of Eq. (4.11) is not satisfied, then the weights can be normalized using the following formula:

$$\tilde{w}_i = \frac{w_i}{\sum_{j=1}^{m} w_j} \tag{4.12}$$

Each weighting coefficient corresponds to the weight of each criterion and is adjusted according to the importance of the criterion. Every combination of the weighting coefficients corresponds theoretically to a single Pareto optimal solution, thus, performing a series of optimization processes using different weighting coefficients it is possible to generate the full set of Pareto optimal solutions. A basic disadvantage of the method is that the generation of the full Pareto set can be guaranteed only for convex problems.

For realistic applications, the values of the various objective functions can be measured in different units and thus be of different order of magnitude, which can cause problems to the optimization process. Thus it is necessary to normalize every objective function according to the following formula

$$\tilde{f}_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - f_{i,\min}}{f_{i,\max} - f_{i,\min}} \tag{4.13}$$

where $f_{i,\min}$ and $f_{i,\max}$ are the minimum and maximum values that the i-th objective function can take in the design space, respectively. The normalized objective functions $\tilde{f}_i(\boldsymbol{x}) \in [0,1]$ use the same design space as the non-normalized ones.

Optimal solutions generated by LWM

On condition that an exact optimization algorithm is used and all weights are positive, the LWM will only generate Pareto-optimal solutions, a statement that can be easily proved: assume that a feasible design vector $\boldsymbol{a}$ minimizes the weighted sum of Eq. (4.10) for a given weight combination and is *not* Pareto optimal. Then, there is a solution $\boldsymbol{b}$ which dominates $\boldsymbol{a}$, i.e. without loss of generality $f_1(\boldsymbol{b}) < f_1(\boldsymbol{a})$ and $f_i(\boldsymbol{b}) \leq f_i(\boldsymbol{a})$ for $i=2,\ldots,m$. Therefore by multiplying the m equations with the corresponding positive weights and adding the inequality equations we obtain:

$$w_1 f_1(\boldsymbol{b}) + w_2 f_2(\boldsymbol{b}) + \ldots + w_m f_m(\boldsymbol{b}) < w_1 f_1(\boldsymbol{a}) + w_2 f_2(\boldsymbol{a}) + \ldots + w_m f_m(\boldsymbol{a}) \Leftrightarrow \quad (4.14)$$

$$\sum_{i=1}^{m} w_i f_i(\boldsymbol{b}) < \sum_{i=1}^{m} w_i f_i(\boldsymbol{a}) \quad (4.15)$$

which is in contradiction with the assumption that the design vector $\boldsymbol{a}$ minimizes the weighted sum of Eq. (4.10).

Drawbacks of the LWM

In general, there are several major disadvantages of using the weighting method (Diwekar 2008):

i. Its inefficiency arising from the linear combination of objectives.

ii. Its difficulty to control the region of the non-dominated surface on which the decision-maker is heavily favored. For example, a small change in the weighting coefficients may cause big changes in the objective vectors, and dramatically differing weighting coefficients may produce nearly similar objective vectors.

iii. An evenly distributed set of weighting vectors does not necessarily produce an evenly distributed representation of the Pareto set, even if the problem is convex (Das and Dennis 1998). This shows a lack of robustness. Furthermore, all of the Pareto optimal points cannot be found if the problem is non-convex (Miettinen 1999).

4.7.2 Distance Function Method (DFM)

The distance methods (Zeleny 1982) are based on the minimization of the distance between the set of the objective function values and some chosen reference points belonging to the criterion space $\mathcal{Y}_{\mathcal{F}}=\boldsymbol{f}(\mathcal{F})$. This technique has been implemented in structural optimization in the work of Koski (1994). The resulting scalar equivalent of the optimization problem of Eq. (4.1) can be written as

$$\min_{\boldsymbol{x}\in\mathcal{F}} d_p(\boldsymbol{x}) \quad (4.16)$$

where $d_p(\boldsymbol{x})$ is the distance function which can be written as:

$$d_p(\boldsymbol{x}) = \left[\sum_{i=1}^{m} w_i (f_i(\boldsymbol{x}) - z_i)^p \right]^{1/p} \tag{4.17}$$

and p is an integer number. The reference point $\boldsymbol{z}^{id}=[z_1, ..., z_m]^T$ of the objective space, also called *ideal or utopian point* is selected by the designer. A reference point that is frequently used is the following:

$$\boldsymbol{z}^* = \left[f_1^*, \ldots, f_m^* \right]^T \tag{4.18}$$

where f_i^* ($i=1,...,m$) is the optimum solution of the single-objective optimization problem where the i-th objective function is treated as the unique objective. The normalization described in Eqs. (4.11) and (4.12) for the weighting factors w_i is also used. In the special case of $p=\infty$, Eq. (4.16) is transformed into the minimax problem:

$$\min_{\boldsymbol{x}\in F} \max_{i=1}^{m} \left[w_i f_i(\boldsymbol{x}) \right] \tag{4.19}$$

In the special case of $p=1$ the DFM is equivalent to the LWM when the reference point used is the zero $\boldsymbol{z}^{id}=\boldsymbol{0}$, while in the special case of $p=2$ the method is called *Weighted Quadratic Method* (WQM).

4.7.3 Constraint Method (CM)

According to the CM, the original multi-objective problem is replaced by a scalar problem where one of the m objectives, called the main objective (f_k), is chosen as the single objective function and all the m-1 others (additional objectives) are removed into the constraints (Haimes and Hall 1974).

By introducing parameters ε_i ($i=1,...,m$ with $i \neq k$), these m-1 constraints can be written as:

$$f_i(\boldsymbol{x}) \leq \varepsilon_i, i = 1, \ldots, m \text{ with } i \neq k \tag{4.20}$$

Then the resulting single-objective optimization problem can be written as follows:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} f_k(\boldsymbol{x}), \quad \boldsymbol{x} = [x_1, \ldots, x_n]^T, \quad f_k \in \mathbb{R} \\ &\text{Subject to} \\ &\boldsymbol{g}(\boldsymbol{x}) \leq 0, \quad \boldsymbol{g} \in \mathbb{R}^p \\ &\boldsymbol{h}(\boldsymbol{x}) = 0, \quad \boldsymbol{h} \in \mathbb{R}^q \\ &f_i(\boldsymbol{x}) \leq \varepsilon_i, \ i = 1, \ldots, m \text{ with } i \neq k \\ &x_i \in X_i \quad \text{for } i = 1, ..., n \end{aligned} \tag{4.21}$$

The solution to the above SOP corresponds to a single point of the Pareto Front of the original MOP. By using various values of these parameters ε_i, the constraint method gives the opportunity to obtain the full set of the Pareto optimum solutions. This technique is

not biased towards convex portions of the Pareto Front, as it is able to obtain solutions associated with non-convex parts of the trade-off curve also.

4.8 Evolutionary Algorithms for solving multi-objective optimization problems

The standard methods for solving multi-objective optimization problems, discussed in Section 4.7, have been used in the past with mathematical programming optimization algorithms where at each optimization step one design point was examined as an optimum design candidate. In order to locate the set of Pareto optimum solutions, a number of optimization runs have to be executed. Recently, *Evolutionary Algorithms* (EAs) have established as an alternative to standard methods for handling MOPs, through which large search spaces can be handled, multiple alternative trade-offs can be generated in a single optimization run and the difficulties encountered by the standard methods can be avoided.

EA optimizers work simultaneously with a population of design points, instead of a single design point, which constitute a population of optimum design candidates that search simultaneously in the space of design variables. Due to this characteristic, methods based on EAs have a great potential in finding the full Pareto front in a single optimization run. Since the early 1990s a number of researchers have suggested the use of EAs in multi-objective optimization problems (Coello Coello 2000; Fonseca and Fleming 1993; Fonseca and Fleming 1995; Horn et al. 1994; Marler and Arora 2004; Schaffer 1984; Zitzler 1999).

In the absence of preference information, none of the corresponding trade-offs can be said to be better than the others. On the other hand, the search space can be too large and too complex, which is the usual case for real-world problems, hence the implementation of gradient based optimizers for this type of problems becomes even more cumbersome. Thus, efficient optimization strategies are required able to deal with the presence of multiple objectives and the complexity of the search space. EAs have several characteristics that are desirable for this kind of problems and most frequently outperform the deterministic optimizers such as gradient based optimization algorithms.

4.9 Evolution Strategies combined with the Linear Weighting Method

In the implementation where the weighting method is used in order to generate a set of Pareto optimal solutions, the optimization procedure initiates with a set of parent design vectors needed by the ES optimizer and a set of weighting coefficients for the combination of all objectives into a single scalar parameterized objective function. These weighting coefficients are not set by the designer but are being systematically varied by the optimizer after a Pareto optimal solution has been achieved.

There is an outer loop which systematically varies the parameters of the parameterized objective function, that is called *decision making loop*. The inner loop is the classical ES

procedure, starting with an initial set of parent vectors. If any of these parent vectors gives an infeasible design then it is modified until it becomes feasible. Subsequently, the offspring are generated and checked whether they are in the feasible region. According to the $(\mu+\lambda)$ selection scheme, in every generation the values of the objective function of the parent and the offspring vectors are compared and the worst vectors are rejected, while the remaining ones are considered to be the parent vectors of the new generation. On the other hand, according to the (μ,λ) selection scheme only the offspring vectors of each generation are used to produce the new generation. This procedure is repeated until the chosen termination criterion is satisfied. The number of parents and offspring involved affects the computational efficiency of the multi-membered ES scheme discussed in this work. The ES algorithm combined with the LWM is shown in Figure 4.3.

Outer loop - Decision making loop

Set the parameters w_i of the parameterized objective function

Inner loop - ES loop

1. *Selection step*: selection of $\boldsymbol{x}^i$ $(i=1,\ldots,\mu)$ parent vectors of the design variables
2. *Analysis step*
3. *Evaluation of parameterized objective function*
4. *Constraints check*: all parent vectors become feasible
5. *Offspring generation*: generate $\boldsymbol{x}^j$ $(j=1,\ldots,\lambda)$ offspring vectors of the design variables
6. *Analysis step*
7. *Evaluation of the parameterized objective function*
8. *Constraints check*: if satisfied continue, else change $\boldsymbol{x}^j$ and go to *step 5*
9. *Selection step*: selection of the next generation parents according to $(\mu+\lambda)$ or (μ,λ) selection schemes
10. *Convergence check*: if satisfied stop, else go to *step 5*

End of Inner loop

End of Outer loop

Figure 4.3 The ES algorithm combined with the Linear Weighting Method.

4.10 Proposed algorithms for evolutionary multi-objective optimization

Two evolutionary algorithms for multi-objective optimization are proposed in the present thesis:

i. The non-dominant *Cascade Evolutionary Algorithm* (CEA);

ii. The *Evolution Strategies for Multi-objective Optimization* (ESMO) algorithm.

4.10.1 The non-dominant Cascade Evolutionary Algorithm

In the present thesis the idea of cascading is implemented within the Evolution Algorithm (EA) context for solving multi-objective structural optimization problems. In particular, the CEA method is employed for the solution of the sequence of parameterized single objective optimization problems. The proposed method is based on the idea of decomposing the multi-objective optimization problem into a series of *nruns* single objective sub-problems which are solved concurrently with the CEA method since the subproblems are independent between each other. CEA consists of *csteps* optimization stages, each of which employs the same optimizer. In order to differentiate the search paths followed by the same optimization algorithm during the cascade stages, the initial conditions of individual optimization runs are suitably controlled by using at each stage:

i. A different initial design (each stage initiates from the solution of the previous stage);

ii. A different seed for the random number generator of the EA process.

The cascade procedure is terminated when no more improvement on the optimum solution vector can be attained over a small number of optimization stages.

In this work the non-dominant Cascade Evolutionary Algorithm is further enhanced with the augmented Tchebycheff metric for solving the multi-objective optimization problem (Lagaros et al. 2005c). This methodology is abbreviated as CEATm$(\mu+\lambda)_{nruns,csteps}$, where μ, λ are the number of the parent and offspring vectors used in the ES optimization strategy, *nruns* is the number of independent CEA runs and *csteps* is the number of cascade stages employed.

The proposed optimization scheme can easily be applied in two parallel computing levels, an external and an internal one. As it was mentioned earlier the multi-objective optimization problem is converted into a series of single objective optimization problems. The solution of each subproblem can be performed concurrently constituting the *external parallel computing level*. On the other hand, the utilization of the natural parallelization capabilities of the CEA methodology within each independent run defines the *internal parallel computing level*.

The augmented weighted Tchebycheff method belongs to the methods with a priori articulation of the preferences for treating the multi-objective optimization problem and, unlike the linear weighting sum method, can be applied effectively to convex as well as to non-convex problems (Miettinen 1999). The weighted Tchebycheff metric can generate any optimal solution, to any type of optimization problem (Steuer 1986). In order to overcome weakly Pareto optimal solutions, the Tchebycheff method formulates the distance minimization problem as a weighted Tchebycheff problem:

$$\min_{\boldsymbol{x}\in\mathcal{F}} \max_{i=1,\ldots,m} \left\{ w_i \frac{\left| f_i(\boldsymbol{x}) - z_i^* \right|}{f_i(\boldsymbol{x})} + \rho \sum_{i=1}^{m} \frac{\left| f_i(\boldsymbol{x}) - z_i^* \right|}{f_i(\boldsymbol{x})} \right\} \tag{4.22}$$

where ρ is a sufficiently small positive scalar (in this work ρ=0.1), $f_i(\boldsymbol{x})$ is the i-th objective function to be minimized, z_i^* is the utopian value of the i-th objective function corresponding to the optimum value obtained when the problem is dealt with as a single objective optimization problem for the i-th objective function and w_i are the weight parameters for the i-th objective function, which are random numbers uniformly distributed in the interval [0,1]. These weight parameters are updated according to Eq. (4.12) in order to fulfil the requirement of Eq. (4.11).

In order to ensure that the Pareto fronts will be composed by feasible designs all the infeasible ones are rejected. The weight coefficients w_i of Eq. (4.22) are implemented in order to achieve convergence of the CEA runs to different directions of the design space. The Pareto front is computed by taking advantage of the search procedure performed for the independent CEA runs in these directions. The idea is to define the Pareto optimal solution reached so far in certain periods of the optimization procedure. These periods are called *global generations*. A global generation, shown in Figure 4.4, is completed when a population of μ+λ feasible designs is obtained for each sub-problem. Therefore, the temporary Pareto front is updated by performing non-dominant search over the global generation (i.e. over a suit of μ+λ by *nruns* feasible designs) and the previous temporary Pareto set. A local Pareto front is computed in every global generation; the global Pareto front is reached when the optimization procedure converges.

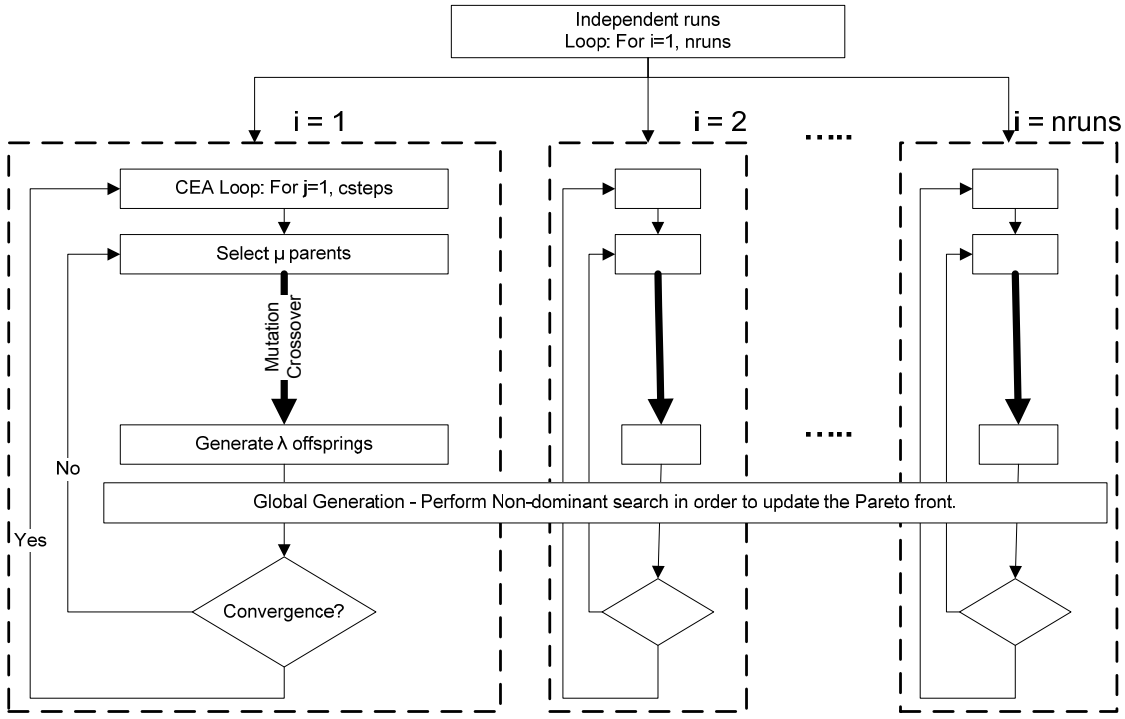


Figure 4.4 The non-dominant Cascade Evolutionary Algorithm.

A non-dominant search is performed in the context of the CEA and the Tchebycheff metric in the sense that all non-dominated solutions attained so far are kept in a set

called temporary Pareto set. It was mentioned before that the multi-objective optimization problems are decomposed into *subproblems* which are solved with independent runs (*nruns* in total) of the CEA methodology. Each subproblem is independent from the other and therefore all subproblems can be dealt with simultaneously. Furthermore, in every global generation a non-dominant search is applied for updating the temporary Pareto set. The global generation is achieved when all local generations of the independent CEA runs are completed. According to this procedure in every global generation a local Pareto front is produced which approaches the global one. A schematic description of the method is given in Figure 4.4. The basic steps inside an independent run of the multi-objective algorithm, as adopted in this study, are shown in Figure 4.5.

Independent run *i*, *i*=1,...,*nrun*

Generate the weight coefficients $w_{i,j}$ (j=1, ...,m) of the Tchebycheff metric. Check if the requirement of Eq. (4.11) is fulfilled, if not change the weight parameters using Eq. (4.12).

CEATm *loop*

Initial generation:

1a. *Generate* $\boldsymbol{x}_k$ (k=1,...,μ) vectors.

1b. *Structural analysis step*

1c. *Evaluation of the Tchebycheff metric*, Eq. (4.22).

1d. *Constraint check*: If satisfied $k=k+1$ else $k=k$. Go to *step 1a*.

Global non-dominant search: check if global generation is accomplished. If yes, then non-dominant search is performed, else wait until global generation is accomplished.

New generation:

3a. *Generate* $\boldsymbol{x}_\ell$ (ℓ=1,...,λ) vectors.

3b. *Structural analysis step*

3c. *Evaluation of the Tchebycheff metric*, Eq. (4.22).

3d. *Constraint check*: If satisfied $\ell=\ell+1$ else $\ell=\ell$. Go to *step 3a*.

Selection step: selection of the next generation parents according to (μ+λ) or (μ,λ) scheme.

Global non-dominant search: Check if global generation is accomplished. If yes, then non-dominant search is performed, else wait until global generation is accomplished.

Convergence check: If satisfied stop, else go to *step 5*.

End of CEATm loop

End of Independent run *i*

Figure 4.5 The CEATm algorithm's steps.

4.10.2 ESMO algorithm

Instead of using scalarizing functions in order to convert the multi-objective optimization problem into a series of single-objective optimization problems, the purpose in using an ES optimizer is to find the full Pareto front with one optimization run, by working with a population of design points that search the space of design variables simultaneously. In order to implement ES for solving multi-objective optimization problems, some modifications have to be introduced to the standard method which was initially designed to search for a single optimum. For this purpose two issues need to be considered:

i. The implementation of fitness selection taking into account all the different objective functions, in order to guide the search towards the Pareto front curve;

ii. The maintenance of a diverse population in order to prevent premature convergence and to achieve a well distributed and well spread Pareto front curve.

In the present thesis an *Evolution Strategies for Multi-objective Optimization* (ESMO) algorithm is proposed for solving the optimization problem. ESMO consists of the standard ES algorithm which has been modified to take into account the issues discussed above (Papadrakakis et al. 2002a). The modifications introduced are presented in the following sections.

Fitness selection

In a single-objective optimization problem the selection operator is based on the value of the single fitness function. In the case of optimization problems with multiple objectives, the selection operator has to be modified in order to allow for multiple objectives. In the present work, the fitness function switches randomly between the objective functions (Schaffer 1984) during the selection phase instead of combining them into a scalarizing function. A random objective function criterion is used to decide whether an individual will be part of the next generation.

Preserve diversity of the population

The goal of fitness sharing is to distribute the population over a number of different peaks in the search space, with each peak receiving a fraction of the population in proportion to the height of that peak. In order to preserve diversity in the population, fitness sharing is implemented (Horn et al. 1994). Sharing calls for the degradation of an individual's objective fitness by a niche count calculated for that individual. According to fitness sharing the fitness values of an individual are degraded depending on the number of the individuals located in its neighborhood, while stable subpopulations are also maintained. This procedure is employed in order to avoid converging to a single optimum. Each subpopulation has its own optimum, which is also called a niche. Fitness sharing causes multiple optimum designs to co-exist in the population. The shared fitness $\tilde{f}_i(\boldsymbol{x})$ of an objective function $f_i(\boldsymbol{x})$ is equal to its initial fitness degraded using the

design's niche count. This degradation is obtained by simply dividing the objective fitness by the niche count to find the shared fitness:

$$\tilde{f}_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x})}{\sum_{h=1}^{\mu\text{-}1} f_{\text{share}}(\boldsymbol{x}, \boldsymbol{x}^h)} \tag{4.23}$$

$$f_{\text{share}}(\boldsymbol{x}, \boldsymbol{x}^h) = \begin{cases} 1 - \left(\dfrac{d(\boldsymbol{x}, \boldsymbol{x}^h)}{\sigma_{\text{share}}} \right) & \text{if} \quad d(\boldsymbol{x}, \boldsymbol{x}^h) < \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases} \tag{4.24}$$

$$d(\boldsymbol{x}, \boldsymbol{x}^h) = \left\| f(\boldsymbol{x}) - f(\boldsymbol{x}^h) \right\| \tag{4.25}$$

where μ is the size of the population, $d(\boldsymbol{x},\boldsymbol{x}^h)$ is a distance function defining the distance of the design $\boldsymbol{x}$ from another member of the population, $\boldsymbol{x}^h$, in the objective space.

To estimate the value of σ_{share}, the following expression is used

$$\sigma_{\text{share}} = \frac{r}{\sqrt[n]{q}} \tag{4.26}$$

where r is the volume of an n-dimensional hypersphere of radius σ_{share}, and q is the number of peaks that the algorithm is asked to locate. In this study q is taken equal to 20.

The sum over the μ-1 members of the population in the denominator of Eq. (4.23) defines the niche count of design $\boldsymbol{x}$. The niche count is an estimate of how crowded the neighborhood (niche) of the individual is. It is calculated over all individuals in the current population. Depending on whether the distance function $d(\boldsymbol{x},\boldsymbol{x}^h)$ works on the phenotypical or the genotypical space, phenotypical or genotypical sharing is defined. In phenotypical sharing the distance between two members of the population $\boldsymbol{x}$ and $\boldsymbol{r}$ is measured with their Euclidean distance in the n-dimensional space, where n refers to the number of design variables, as follows:

$$d(\boldsymbol{x}, \boldsymbol{r}) = \sqrt{\sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{r}_i)^2} \tag{4.27}$$

ESMO algorithm pseudocode

The steps of the ES algorithm for Multi-objective Optimization can be summarized as shown in the flowchart of Figure 4.6.

1. *Initialization*: generate the μ parent vectors of the first generation ($\boldsymbol{x}^i$, $i=1,..,\mu$)
   a. *Design step*: structural analysis - constraints check (Generate μ feasible parent vectors)
   b. *Evaluate the objective functions:* calculate f_1 and f_2
2. *Non-dominated search*: obtain a temporary Pareto front curve
3. *Offspring generation*: generate the λ offspring vectors of the g-th generation ($\boldsymbol{x}_j$, $j=1,...,\lambda$)
   a. *Design step*: structural analysis - constraints check (generate λ feasible offspring vectors)
   b. *Evaluate the objective functions*: calculate f_1 and f_2
4. *Fitness sharing*: all the members of the parent and offspring populations are subjected to fitness sharing (Eqs. (4.23) and (4.24))
5. *Selection step*: select the parent population of the next generation from the parent and offspring populations of the current generation (see fitness selection section).
6. *Non-dominated search*: obtain a new temporary Pareto front set
7. *Convergence check*: if satisfied stop, else go back to step 3.

Figure 4.6 The ESMO algorithm's steps.

Non-dominated search is used in order to find the Pareto optimal set in the current population. The death penalty method is employed for handling the constraints. If a design is not feasible then it is rejected and a new design is generated by means of the recombination and mutations operators (Lagaros et al. (2004)). Details on constraint handling techniques can be found in Section 3.11.4.

4.11 Particle Swarm Optimization for multi-objective problems

Given the similarities of the *Particle Swarm Optimization* (PSO) method with Evolutionary Algorithms and the promising results reported in the literature comparing PSO to EA techniques for single-objective optimization problems, a transfer of PSO to multi-objective problems seems a natural progression. Recently, in 2002 this progression occurred, with a number of different studies published on *Multi-objective Particle Swarm Optimization* (MOPSO) (Coello Coello and Lechuga 2002; Fieldsend and Singh 2002; Hu and Eberhart 2002; Parsopoulos and Vrahatis 2002).

Although most of these studies were generated in tandem, each of them implements MOPSO in a different fashion. The research in the field of multi-objective PSO is an open field, continuing with great interest (Abido 2007; Omkar et al. 2008; Parsopoulos et al. 2004; Reyes-Sierra and Coello Coello 2006; Vlachogiannis and Lee 2009). In the present thesis, the PSO algorithm has been only applied to single objective optimization problems. The application of the method to multi-objective problems will be one of the next research steps, as will be described in detail in Section 9.3 (Future work).

Chapter 5

5 Neural Networks

5.1 Introduction

5.1.1 Historical background

The first wave of interest in *Artificial Neural Networks*, also known as *connectionist models* or *parallel distributed processing models* emerged after the introduction of simplified neurons by McCulloch and Pitts (1943). These neurons were presented as models of biological neurons and as conceptual components for circuits that could perform computational tasks. The interest in the field was reduced and funding was redirected to other scientific fields when Minsky and Papert published the book 'Perceptrons' (Minsky and Papert 1969), illustrating the deficiencies of perceptron models. Only a few researchers continued their work in NNs, until the interest in the field re-emerged after some important theoretical results were attained in the early 1980s (most notably the discovery of error back-propagation) and new hardware developments increased the processing capacities. This renewed interest is reflected in the number of scientists, the amounts of funding, the number of large conferences and the number of journals associated with NNs lately (Krose and van der Smagt 1996).

Artificial neural networks can be most adequately characterized as *computational models* with particular properties such as the ability to adapt or learn, to generalize, or to cluster or organize data, and which operation is based on parallel processing. However, many of the abovementioned properties can be also attributed to other (non-neural) models. The intriguing question is to which extent the neural approach proves to be better suited for certain applications than existing models. Thus, NNs can be generally considered as an alternative computational scheme.

5.1.2 Biological Neural Networks

In neuroscience, a (biological) neural network describes a population of physically interconnected neurons or a group of disparate neurons whose inputs or signaling targets de-

fine a circuit. A neuron consists of a soma (cell body), axons (output connection, that send signals), and several dendrites (input connections, that receive signals). A synapse connects an axon to a dendrite. Communication between neurons often involves an electrochemical process. Given a signal, a synapse might increase (excite) or decrease (inhibit) electrical potential. A neuron fires when its electrical potential reaches a threshold. Learning might occur by changes to synapses. If the sum of the input signals surpasses a certain threshold, the neuron sends an action potential at the axon hillock and transmits this electrical signal along the axon.

A single neuron is very slow compared to a modern computer processor, as it takes about one millisecond (10^{-3}s) to react to an impulse, while a 1GHz processor performs one instruction in a nanosecond (10^{-9}s). However, if we take into account that a human brain has roughly 100 billion (10^{11}) simultaneously functioning neurons and about 100 trillion (10^{14}) synapses, then we can safely presume that a human brain has great computing power. Figure 5.1 depicts a biological neuron (the cell body, or *Soma*) and its connections. Together they comprise the basic building blocks for biological brains.

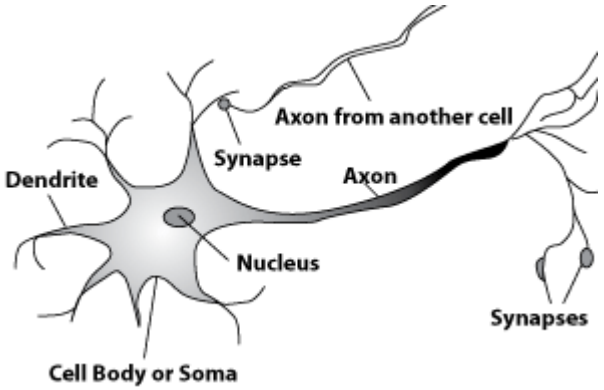


Figure 5.1 A biological neuron.

The first rule of neuronal learning was described by Hebb (1949), named *Hebbian learning*. The theory is commonly evoked to explain some types of associative learning in which simultaneous activation of cells leads to pronounced increases in synaptic strength. Hebbian pairing of pre-synaptic and post-synaptic activity can substantially alter the dynamic characteristics of the synaptic connection and therefore facilitate or inhibit signal transmission.

5.1.3 Artificial Neural Networks

The development of artificial neural networks was initially inspired and motivated by insights into how biological brains – and in particular mammalian brains – function. It was found that mammalian brains learn as connections between neurons are strengthened – the result of electrochemical processes triggered by external or internal stimuli (experiences). As in biological systems, learning involves adjustments to the synaptic connections that exist between the neurons. NNs, like human beings, learn by example.

Although parallels with biological systems are often described, there is still so little known (even at the lowest cell level) about biological systems, that the models that are being used for artificial neural systems seem to introduce an oversimplification of the 'biological' models. The real, biological nervous system is highly complex and includes some features that may seem superfluous based on an understanding of artificial networks.

In the first work on the processing of neural networks (Lettvin et al. 1959) it was shown theoretically that networks of artificial neurons could implement logical, arithmetic, and symbolic functions. Simplified models of biological neurons were set up, now usually called perceptrons or artificial neurons. These simple models accounted for neural summation, i.e. potentials at the post-synaptic membrane would sum in the cell body. Later models also provided for excitatory and inhibitory synaptic transmission.

Artificial Neural Networks are made up of fully or partially interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. *Artificial Intelligence* (AI) and *cognitive modeling* try to simulate some properties of neural networks. While similar in their techniques, AI has the aim of solving particular tasks, while cognitive modeling aims to build mathematical models of biological neural systems.

In the AI field, artificial neural networks have been trained to perform complex functions in various scientific fields and have been applied successfully to identification, classification, simulation, inverse simulation, speech recognition, pattern recognition, image analysis and adaptive control, and also in order to construct software agents (in computer games) or autonomous robots. Most of the currently employed artificial neural networks for artificial intelligence are based on statistical estimation, optimization and control theory. In the present thesis, the term *Neural Network* (NN) will refer to an artificial Neural Network, as opposed to a biological neural network.

5.2 Soft computing as opposed to Hard computing

5.2.1 The concept of computing

Computing takes place in human designed personal computers as well as biologically evolved computers (brains). All computers perform the same fundamental task of information processing. Information to be processed in computing is normally contained in some data. Desktop and laptop computers are designed to do *hard computing*. On the other hand, brains are also computers, that have evolved to do *soft computing*. Desktops and laptops can be used today in order to simulate soft computing.

5.2.2 Hard computing

Hard computing methods are based on mathematics. They inherit the properties of mathematically based problem solving methods that are employed in science and engineering. The properties of mathematically based hard computing methods are:

i. **Precision**: Using a personal computer in order to solve a mathematical problem, the solution can be given in any desired accuracy, based on the program's settings.

ii. **Universality**: In practical problems, one is normally interested only in a range of variables. Mathematically based hard computing methods provide universal solutions, that are valid for all the possible ranges of variables.

iii. **Functional uniqueness**: Mathematical functions are unique. For example there is only one $\cos(x)$ and it is valid for all the possible values of x.

Hard computing methods, such as the *Finite Element Method* (FEM) have been extremely useful in engineering problem solving. However, it is important to recognize some fundamental limitations that are the consequences of the properties that hard computing methods inherit from mathematics:

i. **Lack of robustness**: For instance, a *Finite Element Analysis* (FEA) will likely fail if a portion of the computer program is removed. Many solution methods also lack robustness. This is a direct consequence of the precision requirement in hard computing.

ii. **Suitability for forward problems**: Mathematically based hard computing methods are inherently suitable for solving forward problems. Yet, they lack the capability for directly solving inverse problems.

iii. **Relatively slow operation**: Operations in hard computing methods are sequential and therefore relatively slow.

5.2.3 Soft computing

Soft computing methods are inspired by the computing and problem solving strategies in nature. They inherit their properties from nature's problem solving methods and they are fundamentally different from the mathematically based hard computing problem solving methods that are routinely used in engineering.

Problem solving in nature

The human brain is a massively parallel computer that has evolved through time in nature. It is built and operates very differently than personal computers. Large number of neurons that perform simple tasks are interconnected, giving the brain complex properties. The brain has evolved in order to solve problems that are important in the survival of species.

Human beings can normally recognize up to about 200 faces and voices. The tasks of face and voice recognition that our brains perform so well are *inverse problems* that have to be solved in *real time* and *robustly*. The massively parallel nature of the brain is the reason that these difficult inverse problems are solved in real-time and robustly. In fact, one may surmise that the brain has evolved as a massively parallel system, partly because it has to solve the problems this way for the survival of species.

The characteristics of soft computing methods for problem solving in nature are:

i. **Operating with robustness**: Face and voice recognition occurs in a robust way. For instance, humans can recognize partially covered faces.

ii. **Directly solving inverse problems**: Our brains solve inverse problems. Nature solves inverse problem through evolution. Face and voice recognition or an animal recognizing its prey is a typical inverse problem that has to be solved directly.

iii. **Operating and providing response in real time**: In nature, there is no time for slow, sequential algorithmic thinking and problems has to be solved in real time.

While hard computing methods rely on mathematics, soft computing methods rely on nature's problem solving strategies, such as: (i) Learning; (ii) Reduction in disorder; and (iii) Random search and gradual improvement. The main properties of soft computing methods are:

i. **Imprecision tolerance**: Precision is not that important. Nature's problem solving strategies have evolved with *imprecision tolerance*.

ii. **Non-universality**: Universality is also not important. Face and voice recognition methods in human brain have evolved to recognize a limited number of faces and voices; not all the possible faces and voices. *Non-universality* is an important property or nature's problem solving methods.

iii. **Functional non-uniqueness**: While the details of neural connections in brains are different, they can perform similar tasks with imprecision tolerance. This amounts to *functional non-uniqueness*.

Understanding the differences between hard and soft computing is very significant in full utilization of the potential of the biologically inspired soft computing methods. It is very important that soft computing methods are not treated as another form of mathematically based problem solving methods.

Inverse problems in engineering

The vast majority of engineering problems are inverse problems. For instance, the fundamental engineering tasks of structural modeling and design are inverse problems. Yet, we do not pose these issues as inverse problems, because our mathematically based engineering problem methods are not suitable for solving inverse problems. Such problems are often solved by repeated application of forward solutions. Soft computing methods

inherit the capability for solving inverse problems directly from nature's problem solving strategies.

Neural Networks as soft computing tools

Neural Networks belong to soft computing methods, inspired by the massively parallel structure and operation of the brain, inheriting the main properties of nature's problem solving strategies. In that sense, they are fundamentally different than the mathematically based hard computing methods. An implication of the above observations is that soft computing methods such as neural networks, have also the potential of solving inverse problems robustly and in real-time.

5.3 Neural Networks characteristics

Neural networks exhibit a remarkable ability to derive meaning from complicated or imprecise data, extract patterns and detect trends that are too complex to be noticed by either humans or other computational techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. NN characteristics are the following:

1. **Adaptive learning**: An ability to learn how to do tasks based on the data given for training or initial experience.
2. **Self-Organization**: A NN can create its own organization or representation of the information it receives during learning time.
3. **Real Time Operation**: NN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. **Fault Tolerance via Redundant Information Coding**: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even after major network damage.

Neural networks take a different approach to problem solving than conventional computing techniques. Conventional computing uses an algorithmic approach to problem solving, i.e. the computer follows a set of instructions in order to solve a problem. The whole process is totally predicable. Unless the specific steps are known, the computer cannot solve the problem. That restricts the problem solving capability of conventional computing to problems that we already understand and know how to solve.

Yet, computers would be so much more useful if they could do things that we don't exactly know how to do. Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The

examples must be selected carefully in order for the network to function properly and provide useful results.

5.4 Activation functions

Most units in a NN transform their net inputs by using a scalar-to-scalar function called *activation function* or *transfer function*, yielding a value called the unit's activation. Except possibly for output units, the activation value is fed to one or more other units. Activation functions with a bounded range are often called squashing functions. Some of the most commonly used activation functions are (Fausett 1994):

5.4.1 Identity function

The output of the identity function is given by

$$a = f(n) = n \tag{5.1}$$

It is obvious that the input units use the identity function. Sometimes a constant is multiplied by the net input to form a linear function. The graphical representation of the identity function is given in Figure 5.2(a).

5.4.2 Binary step function

The binary step function is also known as *Threshold function*, *Heaviside function* or *Hard limit function*. The output of this function is limited to one of the two values

$$a = f(n) = \begin{cases} 0, & if\ n < 0 \\ 1, & if\ n \geq 0 \end{cases} \tag{5.2}$$

This kind of function, often used in single layer networks, is non-continuous and non-differentiable at point $n=0$. The graphical representation of the binary step function is shown in Figure 5.2(b).

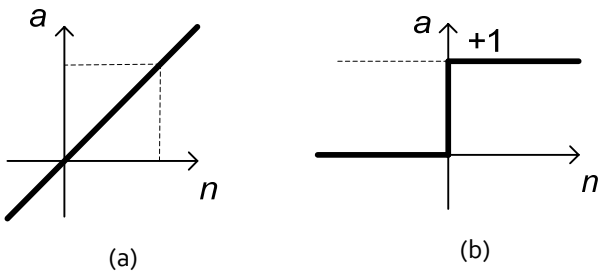


Figure 5.2 (a) Identity function, (b) Hard limit function.

5.4.3 Binary sigmoid function

The output of the binary sigmoid function and its derivative are given by

$$a = f(n) = \frac{1}{1+e^{-n}} \tag{5.3}$$

$$f'(n) = \frac{e^{-n}}{(1+e^{-n})^2} = a(1-a) \tag{5.4}$$

This function is especially advantageous for use in NNs trained by back-propagation, because it is easy to differentiate, and thus can dramatically reduce the computation burden for training. It yields output values in the interval [0,1], while its derivative yields output values in the interval [0,0.25]. The graphical representations of the binary sigmoid function and its derivative are given in Figure 5.3.

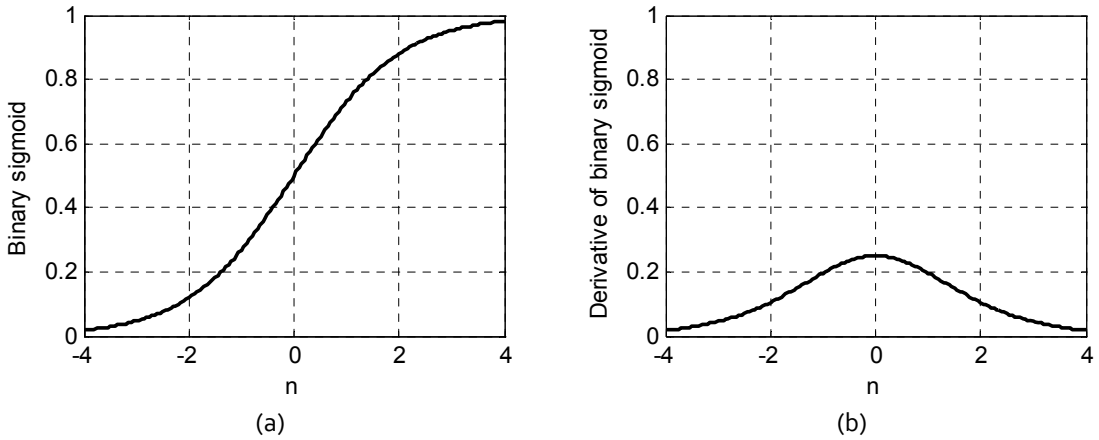


Figure 5.3 (a) Binary sigmoid function, (b) Its derivative.

5.4.4 Bipolar sigmoid function

The output of the bipolar sigmoid function and its derivative are given by

$$a = f(n) = \frac{1-e^{-n}}{1+e^{-n}} \tag{5.5}$$

$$f'(n) = 2\frac{e^{-n}}{(1+e^{-n})^2} = \frac{1}{2}(1+a)(1-a) \tag{5.6}$$

This function has similar properties with the sigmoid function. It yields output values in the interval [-1,1], while its derivative yields output values in the interval [0,0.5]. The graphical representations of the sigmoid function and its derivative are given in Figure 5.4.

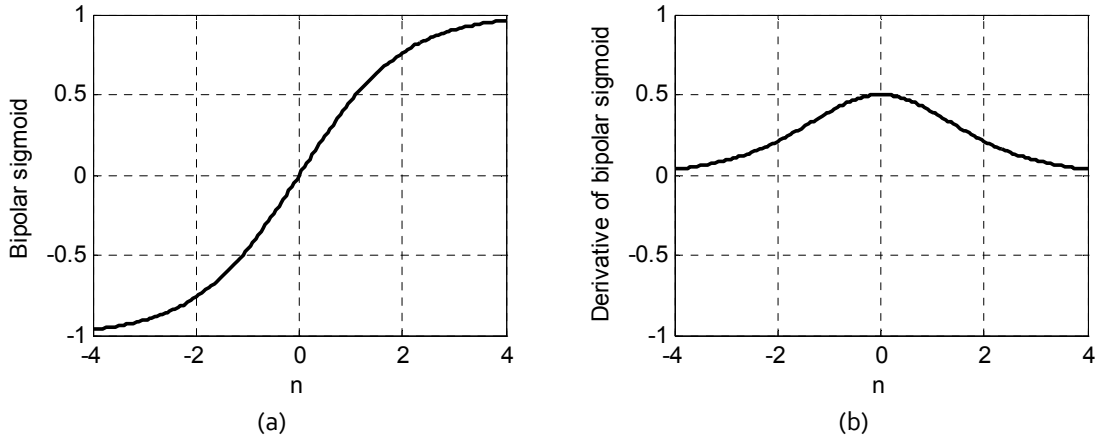


Figure 5.4 (a) Bipolar sigmoid function, (b) Its derivative.

5.4.5 Hyperbolic tangent function

The output of the hyperbolic tangent function and its derivative are given by

$$a = f(n) = \frac{e^{2n} - 1}{e^{2n} + 1} = \lambda \cdot \tan(\lambda x) \tag{5.7}$$

$$f'(n) = 4\frac{e^{2n}}{(e^{2n} + 1)^2} = 1 - a^2 \tag{5.8}$$

This function has similar properties with the binary sigmoid and the bipolar sigmoid functions. It yields output values in the interval [-1,1], while its derivative yields output values in the interval [0,1]. The graphical representations of the hyperbolic tangent function and its derivative are shown in Figure 5.5.

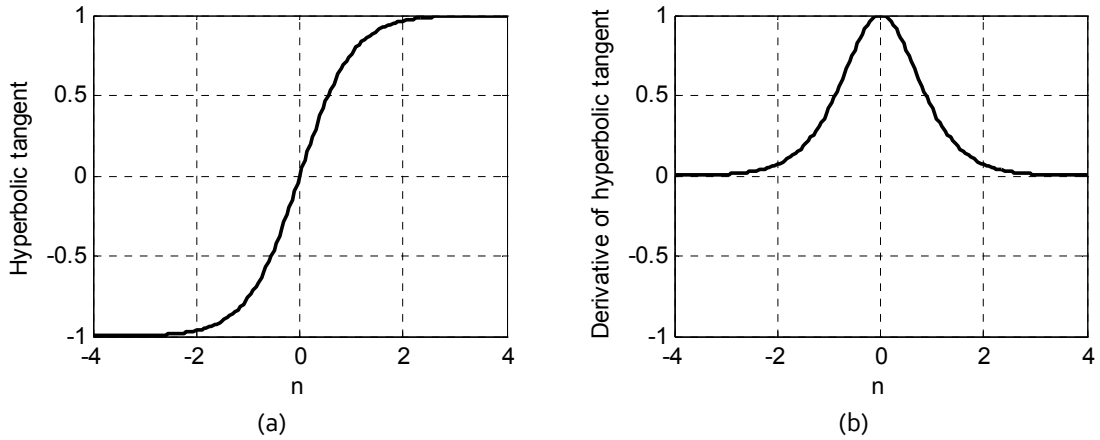


Figure 5.5 (a) Hyperbolic tangent function, (b) Its derivative.

5.4.6 The choice of proper activation functions

Non-linear activation functions for the hidden units are needed in order to introduce non-linearity into the network, as a composition of only linear functions would be again a linear function. However, it is the non-linearity (i.e. the capability to represent non-linear functions) that makes multi-layer networks that powerful. Almost any non-linear function can be used as an activation function, although for back-propagation learning it must be differentiable and it helps if the function is bounded and also if it has an easy to calculate derivative. In any case, the sigmoid functions are the most common choices.

For the output units, activation functions should be chosen to be suited to the distribution of the target values. For binary [0,1] outputs, the sigmoid function is an excellent choice. For continuous-valued targets with a bounded range, the sigmoid functions are again useful, provided that either the outputs or the targets to be scaled to the range of the output activation function. But if the target values have no known bounded range, it is better to use an unbounded activation function, most often the identity function (which amounts to no activation function). If the target values are positive but have no known upper bound, an exponential output activation function can be used.

5.5 Neural Networks elements

Neural networks are composed of simple elements operating in parallel. As in nature, the connections between elements largely determine the network's function. A NN can be trained to perform a particular function by adjusting the values of the connections (weights) between elements. Typically, neural networks are adjusted, or trained, so that a particular input will lead to a specific target output. Figure 5.6 illustrates this situation where the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are needed to train the network.

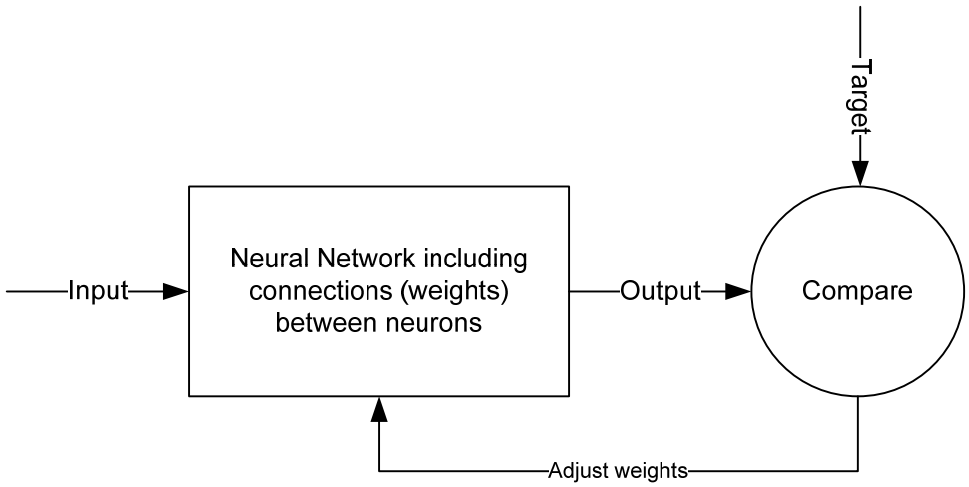


Figure 5.6 Simple Neural Network training flowchart.

NNs exhibit several distinguishing elements or features (Krose and van der Smagt 1996):

- A set of processing units;
- An activation state for each unit, which is equivalent to the output of the unit;
- Connections between the units, each connection being defined by a weight $w_{j,k}$ that determines the effect that the signal of unit j has on unit k;
- A propagation rule, which determines the effective input of the unit from its external inputs;
- An activation function, which determines the new level of activation based on the effective input and the current activation;
- An external input (called bias) for each unit;
- A method for information gathering (learning rule);
- An environment within which the system can operate, provide input signals and, if necessary, error signals.

5.5.1 Simple Neuron with scalar input

A neuron with a single scalar input and a bias appears in Figure 5.7. The specific notation is the one used in Matlab (Demuth et al. 2008).

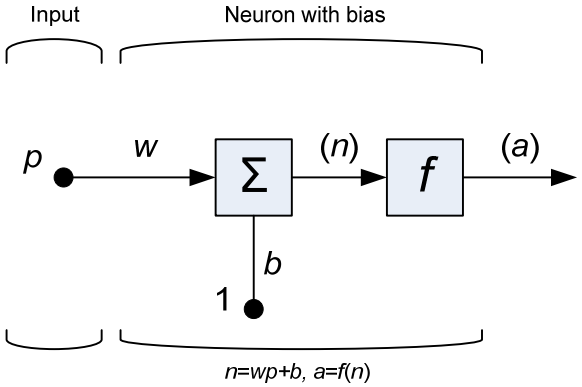


Figure 5.7 A simple neuron with bias.

The scalar input p is transmitted through a connection that multiplies its strength by the scalar weight w to form the scalar product $w \cdot p$. A scalar bias, b, is added to the sum. The sum of the weighted input $w \cdot p$ and the bias b, $n = w \cdot p + b$, is the argument of the *transfer function f*, typically a step function or a sigmoid function, that takes the argument n and produces a scalar output a. Details on the transfer functions that can be used can be found in Section 5.4.

The bias b can be viewed as simply being added to the product $w \cdot p$ as shown by the summing junction, or as shifting the function f to the left by an amount b. The bias is like

a weight, except that it has a constant input of 1. The bias is not an input, yet the constant 1 that drives the bias is an input and must be treated as such. Both *w* and *b* are adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted in order for the network to exhibit the desired behavior. Thus, the network can be trained to do a particular task by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.

5.5.2 Neuron with vector input

A neuron with a single *R*-element input vector is shown below.

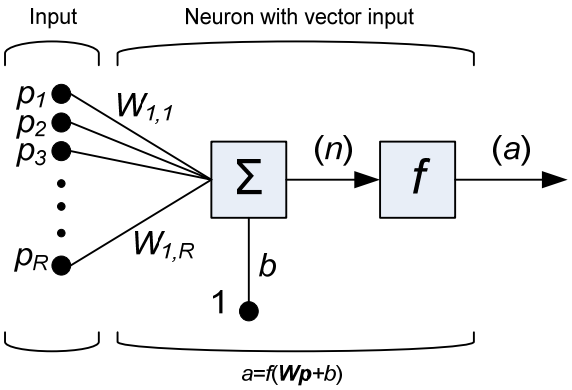


Figure 5.8 A neuron with a single *R*-element input vector.

Here the individual element inputs $p_1, \ldots, p_R$ are multiplied by weights $w_{1,1}, \ldots, w_{1,R}$ and the weighted values are fed to the summing junction. The sum is $\boldsymbol{W}\cdot\boldsymbol{p}$, the dot product of the (single row) matrix $\boldsymbol{W}=[w_{1,1}, \ldots, w_{1,R}]$ and the vector $\boldsymbol{p}=[p_1, \ldots, p_R]^T$. The neuron has a bias *b*, which is summed with the weighted inputs to form the net input *n* which is the argument of the transfer function *f*:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b = \boldsymbol{W}\cdot\boldsymbol{p} + b \tag{5.9}$$

In the above case, $\boldsymbol{p}$ is a column vector ($R\times1$), $\boldsymbol{W}$ is a row vector ($1\times R$) and *b* is a scalar.

5.5.3 A layer of neurons

Two or more of the neurons shown above in Figure 5.8 of Section 5.5.2 can be combined in a layer, and a particular network could contain one or more such layers. A layer includes the combination of the weights, stored in matrix $\boldsymbol{W}$ ($S\times R$), the multiplication and summing operation (here realized as a matrix product $\boldsymbol{W}\cdot\boldsymbol{p}$), the vector bias $\boldsymbol{b}$ ($S\times1$), and the activation (transfer) function *f*, described in Section 5.4. The array of inputs, vector $\boldsymbol{p}=[p_1, \ldots, p_R]^T$, is not included in or called a layer. A one-layer network with *R* input elements and *S* neurons is shown in Figure 5.9.

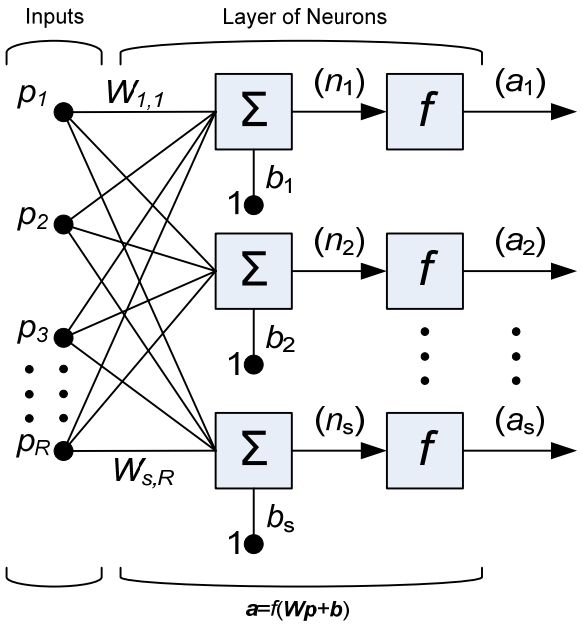


Figure 5.9 A layer of neurons.

In this network, each element of the input vector $\boldsymbol{p}$ is connected to each neuron input through the weight matrix $\boldsymbol{W}$. The i-th neuron has a summer that gathers its weighted inputs and bias to form its own scalar output n_i (i=1, ...,S). The various n_i taken together form a net input vector $\boldsymbol{n}$ (S×1).

$$\boldsymbol{n} = \boldsymbol{W} \cdot \boldsymbol{p} + \boldsymbol{b} \tag{5.10}$$

Finally, the neuron layer outputs form a column vector $\boldsymbol{a}=f(\boldsymbol{Wp}+\boldsymbol{b})$, where function f operates on every element of the input vector $\boldsymbol{Wp}+\boldsymbol{b}$ and the output is also a vector (S×1).

The input vector elements enter the network through the weight matrix $\boldsymbol{W}$. The row indices on the elements of matrix $\boldsymbol{W}$ indicate the destination neuron of the weight, and the column indices indicate which source is the input for that weight. Thus, $w_{j,k}$ denotes the strength of the signal from the k-th input element to the j-th neuron.

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{5.11}$$

Single-layer Perceptron

A single-layer Perceptron or simply *Perceptron* is a layer of neurons with threshold activation function on the output units and biases. The perceptron has a discrete output (+1 or −1). It can be seen as the simplest kind of feedforward neural network: a linear classifier. The Perceptron is a binary classifier that maps its input $\boldsymbol{p}$ (a real-valued vector) to an output value a, a single binary value.

Although the perceptron initially seemed promising, it was eventually proved that perceptrons could not be trained to recognise many classes of patterns. They proved to be only capable of learning linearly separable patterns. In the famous monograph 'Perceptrons' (Minsky and Papert 1969) it was shown that it was impossible a single-layer perceptron to learn an *Exclusive OR* (XOR) function. The authors conjectured (incorrectly) that a similar result would hold for a multi-layer perceptron network. This led to the field of neural network research stagnating for many years, before it was recognised that a feedforward neural network with two or more layers (multilayer perceptron) had far greater processing power than perceptrons with one layer. It took ten more years until neural network research experienced resurgence in the 1980s.

Most perceptrons have outputs of 1 or -1 with a threshold of 0 and there is some evidence that such networks can be trained more quickly than networks created from nodes with different activation and deactivation values. Perceptrons can be trained by a simple learning algorithm that is usually called the *Delta rule*. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent.

It should be noted that in the literature the term *Perceptron* often refers to networks consisting of just one of these units, as opposed to a layer of such units.

5.5.4 Multiple layers of neurons

A network can have several layers of neurons. Each layer has a weight matrix $\boldsymbol{W}$, a bias vector $\boldsymbol{b}$, and an output vector $\boldsymbol{a}$. To distinguish between the weight matrices, output vectors, etc., for each of these layers in the figures, the number of the layer is appended as a superscript to the variable of interest. A two-layer network using the above notation is shown in Figure 5.10.

The network shown in the figure has R inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. The outputs of each intermediate layer are the inputs to the next layer. The layers of a multilayer network play different roles. A layer that produces the network output is called an *output layer*. All other layers are called *hidden layers*. The two-layer network shown in Figure 5.10 has one output layer (layer 2) and one hidden layer (layer 1). Some authors refer also to the inputs as another layer.

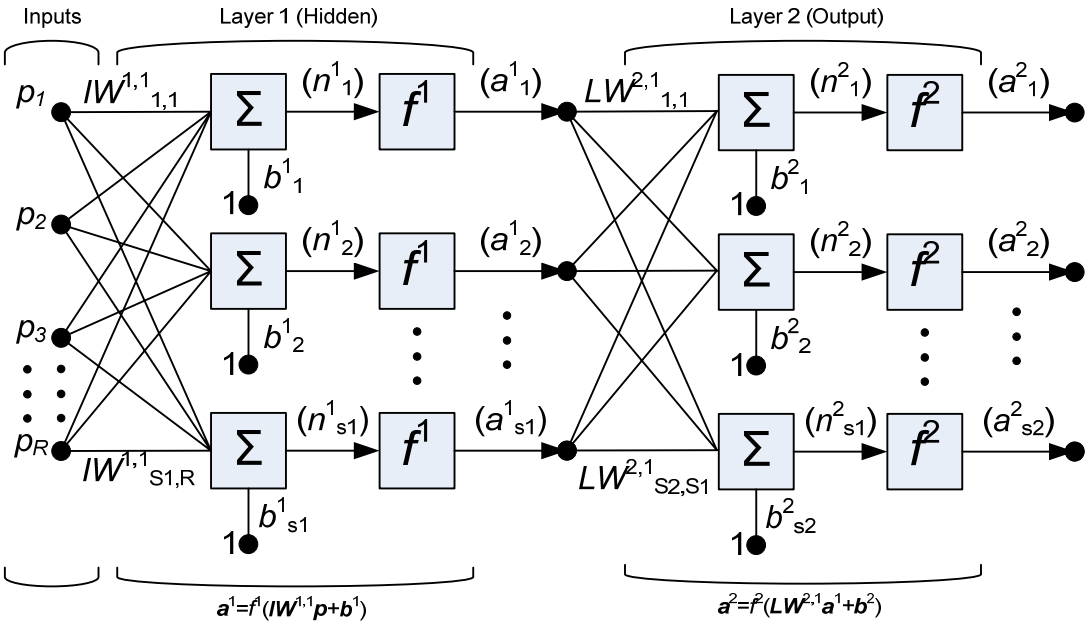


Figure 5.10 A two-layer network.

Multiple-layer networks are quite powerful. The *universal approximation theorem* for neural networks states that the standard multilayer feed-forward network with a single hidden layer that contains finite number of hidden neurons, and a sigmoid activation function are universal approximators on a compact subset of $\mathbb{R}^n$. This means that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer feed-forward NN with just one hidden layer. Kurt Hornik (1991) showed that it is not the specific choice of the activation function, but rather the multilayer feed-forward architecture itself which gives neural networks the potential of being universal approximators. The output units are always assumed to be linear. This kind of two-layer network is used extensively in Back-propagation, as will be discussed in Section 5.9.

Multi-layer Perceptron

A multi-layer Perceptron consists of several layers of single-layer Perceptrons. Figure 5.11 depicts a two-layer perceptron capable of calculating the XOR function. The network has two neurons in the hidden layer (layer 1) and one neuron in the output layer (layer 2). The inputs p_1 and p_2 are binary values (0 or 1). The weights and biases of the network are given in the figure. The network uses the binary step function described in Section 5.4.2 as the transfer function for both layers (f_1 and f_2). The output a^2_1 of the network is also a binary value representing XOR(p_1,p_2).

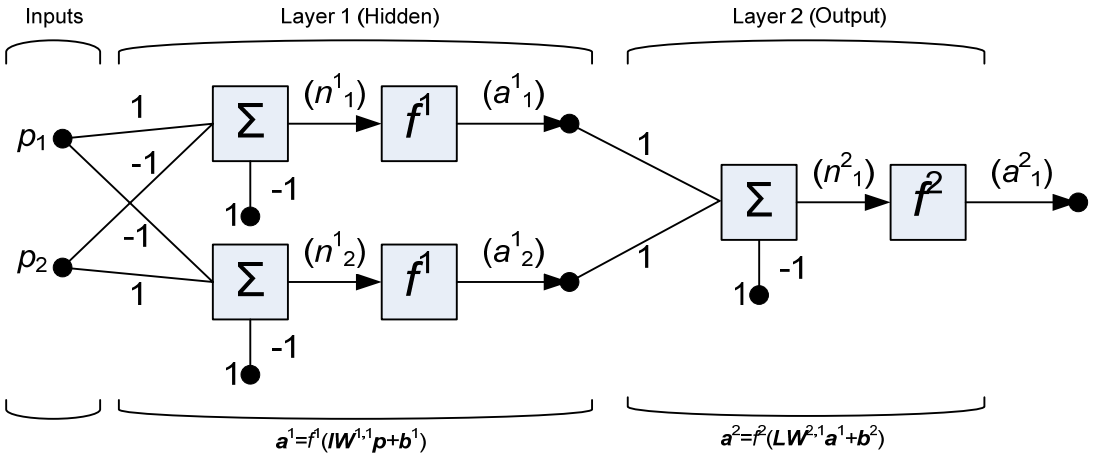


Figure 5.11 A two-layer perceptron capable of calculating the XOR function.

Graphical representations of the OR and XOR functions are given in Figure 5.12 (a) and (b), respectively. Black points indicate a value of "1" while white points indicate a value of "0". It can be seen that the result of the OR function is linearly separable as the diagonal of the triangle of Figure 5.12(a) can divide the points in black and white ones. On the other hand, the result of the XOR function, shown in Figure 5.12(b) is *not* linearly separable, as there is no single line that can divide the black and white points.

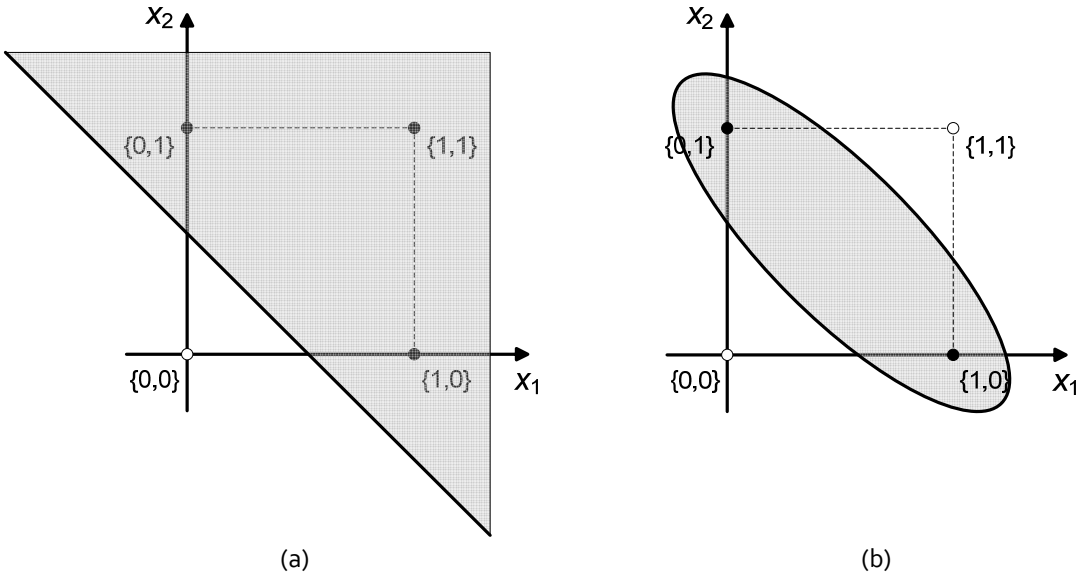


Figure 5.12 Graphical representations of the (a) OR function and (b) XOR function.

The numerical outputs of the OR and XOR functions are given in Table 5.1.

Table 5.1 Output of the OR and XOR functions.

Input	OR Output	XOR Output
{0,0}	0	0
{1,0}	1	1
{0,1}	1	1
{1,1}	1	0

5.5.5 Abbreviated Notation for NNs

When networks with many neurons and layers of many neurons are considered, an abbreviated notation (Demuth et al. 2008) for an individual neuron or a layer of neurons can be used.

Neuron with vector input

The notation of Figure 5.13 is an abbreviated notation for the single neuron with vector input shown in Figure 5.8 of Section 5.5.2. The input vector $\boldsymbol{p}$ is represented by the solid dark vertical bar at the left. The dimensions of $\boldsymbol{p}$ are shown below the symbol $\boldsymbol{p}$ in the figure as Rx1.

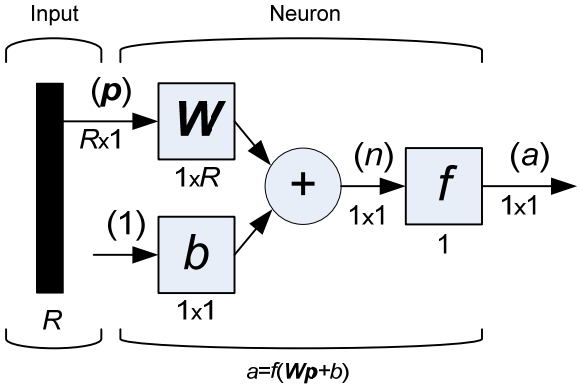


Figure 5.13 Abbreviated notation for a neuron with vector input.

A layer of neurons

The notation of Figure 5.14 is an abbreviated notation for the S neuron R input one-layer network shown in Figure 5.9 of Section 5.5.3. The R length input vector is $\boldsymbol{p}$, $\boldsymbol{W}$ is an $S \times R$ matrix, and $\boldsymbol{b}$ and $\boldsymbol{a}$ are vectors with dimension S.

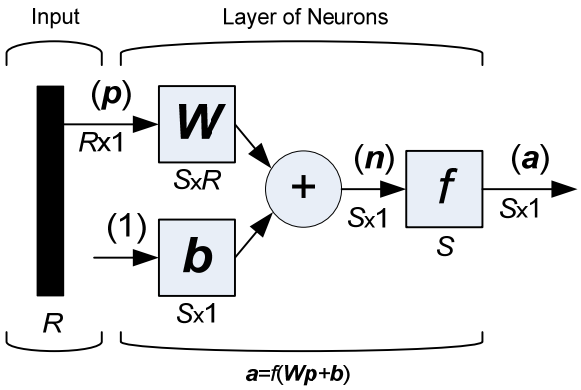


Figure 5.14 Abbreviated notation for a layer of neurons with vector input.

Multiple layers of neurons

The notation of Figure 5.15 is an abbreviated notation for the two-layer network shown in Figure 5.10 of Section 5.5.4.

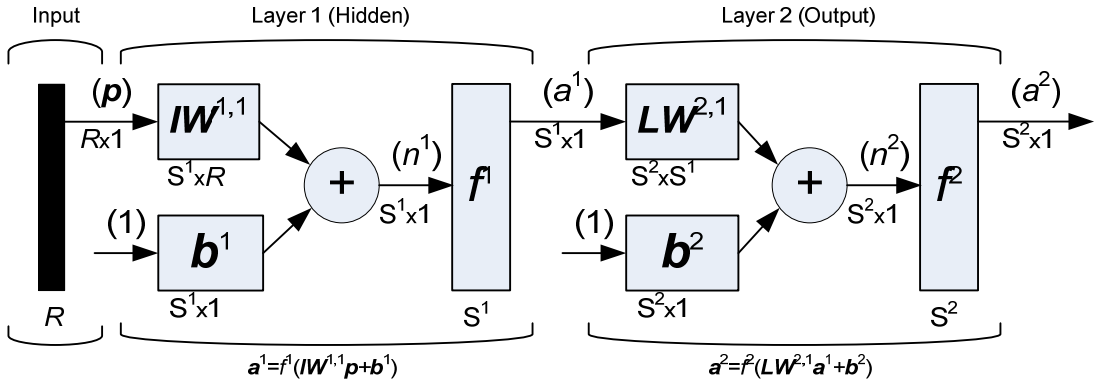


Figure 5.15 Abbreviated notation for a two-layer network.

Multiple Layers of Neurons use layer weight matrices as well as input weight matrices. In the above notation for multiple layers, weight matrices connected to inputs are called input weights (***IW***); weight matrices coming from layer outputs are called layer weights (***LW***). Furthermore, superscripts are used to identify the source (second index) and the destination (first index) for the various weights and other elements of the network. The weight matrix connected to the input vector $\boldsymbol{p}$ is labeled as an input weight matrix $\boldsymbol{IW}^{1,1}$ having a source 1 (second index, which means the first input vector) and a destination 1 (first index, which means the first hidden layer). Elements of layer 1, such as its bias, net input, and output have a superscript 1 as they are associated with the first layer. For instance, weight matrix $\boldsymbol{LW}^{2,1}$ is the matrix that holds the weights of connections from the first layer to the second layer, $\boldsymbol{b}^2$ is the bias vector of the second layer, while $\boldsymbol{a}^2$ is the output of the second (output) layer.

5.6 Network topologies

The topology of a network is defined in general by the number of layers, the number of units per layer, and the interconnection patterns between layers. Based on the pattern of connections, networks are generally divided into two categories (Krose and van der Smagt 1996):

i. Feed-forward networks;

ii. Recurrent networks.

5.6.1 Feed-forward networks

A *feed-forward neural network* is an artificial neural network where the data flow from the input units to the output units is strictly feed-forward, so the information moves in only one direction. The data processing can extend over multiple layers of units, but no feedback connections are present. That is, connections extending from outputs of units to inputs of units in the same layer or previous layers are *not* permitted, thus connections between the units do not form cycles or loops. The feed-forward neural network was the first and arguably simplest type of artificial neural network devised. Feed-forward networks are the main focus of the present thesis. A feed-forward NN is depicted in Figure 5.16, where all connections (lines) have a direction from the left to the right. This can be called a feed-forward 4-3-2-1 NN, the numbers denoting the number of hidden neurons in the input, the two hidden and the output layer (from left to right), respectively.

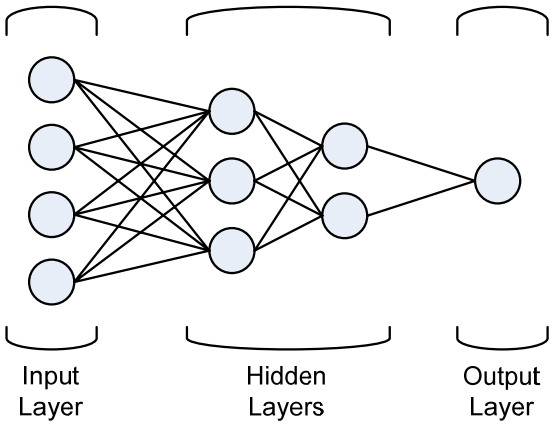


Figure 5.16 A three-layer 4-3-2-1 feed-forward NN.

According to Saarinen et al. (1993) the feed-forward NN is likely to produce ill-conditioned Jacobian matrices due to the bad properties of the activation function used and this type of ill-conditioning is encountered in some applications.

5.6.2 Recurrent networks

Recurrent Neural Networks (RNNs) contain feedback connections. Much of the early work on recurrent networks was pioneered by John Hopfield (1982). In fact, some have argued that it was because of Hopfield's stature as a well-known physicist that neural network research was made respectable again (Anderson and Rosenfeld 1989). Hence, certain configurations of recurrent networks are referred to as Hopfield nets.

The human brain is a recurrent neural network. Contrary to feed-forward networks, the dynamical properties of recurrent networks are important. In some cases, the activation values of the units undergo a relaxation process such that the network will evolve to a stable state in which activation does not change further. In other applications in which the dynamical behavior constitutes the output of the network, the changes of the activation values of the output units are significant.

Figure 5.17 depicts a recurrent NN with two hidden layers, with one recurrent connection from the first hidden layer to itself. Each time a pattern is presented, the unit computes its activation just as in a feed forward network. However its net input now contains a term which reflects the state of the network (the first hidden unit activation) before the pattern was seen. When subsequent patterns are presented, the hidden and output units' states will be a function of everything the network has seen so far. The network behavior is based on its history, and so pattern presentation should be considered as it happens in time.

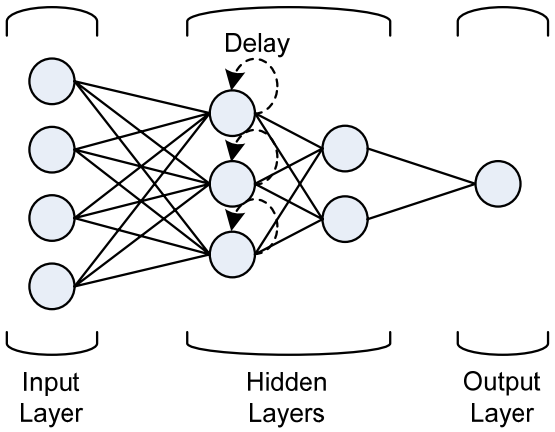


Figure 5.17 A three-layer recurrent NN.

There is growing interest in RNNs for technical applications, as they seem to be more powerful and biologically more plausible than other adaptive approaches such as Hidden Markov Models, feed-forward networks and Support Vector Machines. RNNs have been used successfully in various scientific applications, such as learning formal grammars, speech recognition and music composition.

5.7 Input and Target vectors normalization

The input vectors for a neural network model can be a set of raw data, preprocessed data or a set of parameters. As each individual element of an input vector measures different things, in different units in general, it is obvious that these elements can have great differences in their values, e.g. for an input vector $\boldsymbol{p}$ consisting of 2 elements, values for p_1 measuring a section's moment of inertia may be in the interval [200, 8000] in cm^4 units, while values for p_2 measuring the material's modulus of elasticity may be in the interval $[1.5\times10^8, 3.5\times10^8]$ in kPa units. The same also applies for the target vectors of the neural network model that can also measure different things, in different units, for example the maximum displacement and the maximum stress of a finite element model. For this reason, it is necessary to perform some kind of normalization for the inputs and target vectors of the model, which helps avoid convergence problems, speeds up the learning process for many networks and makes NN training more efficient.

For the back-propagation neural network model, that will be described in detail in Section 5.9, each input should be normalized between 0 and 1 if the activation function used is the standard sigmoid function and between -1 to 1 for the hyperbolic tangent function (Kim 1999). In order to normalize the data, we use some processing functions for the inputs and targets of the NN model. Processing functions transform user input and target data to a form that is easier or more efficient for a network.

Forward processing function

Let Q be the number of patterns for inputs and targets, $\boldsymbol{p}$ be an input vector of dimension R, while $\boldsymbol{t}$ be a target vector of dimension S. The training set $\mathcal{T}$ consists of Q pairs of $\boldsymbol{p}$ and $\boldsymbol{t}$ vectors as follows:

$$\mathcal{T} = \left\{ \{\boldsymbol{p}^1, \boldsymbol{t}^1\}, \ldots, \{\boldsymbol{p}^Q, \boldsymbol{t}^Q\} \right\} \tag{5.12}$$

The following normalization formula is applied for pattern k and dimension i (i=1, ..., R) of the input vector $\boldsymbol{p}$:

$$\tilde{p}_i^k = \frac{p_i^k - p_i^{\min}}{p_i^{\max} - p_i^{\min}} \tag{5.13}$$

where

$$p_i^{\min} = \min\{p_i^k, \quad k = 1, \ldots Q\} \tag{5.14}$$

$$p_i^{\max} = \max\{p_i^k, \quad k = 1, \ldots Q\} \tag{5.15}$$

The above formula maps the range of input values to the range [-1,1]. Similarly, network targets (outputs) can also have associated processing functions. Output processing functions are used to transform user-provided target vectors for network use. The same ap-

proach can be also applied for the target vectors, where the following normalization formula is applied for pattern k and dimension i (i=1,...,S) of the target vector $\boldsymbol{t}$:

$$\tilde{t}_i^k = \frac{t_i^k - t_i^{\min}}{t_i^{\max} - t_i^{\min}} \tag{5.16}$$

where

$$t_i^{\min} = \min\{t_i^k, \quad k = 1,\dots Q\} \tag{5.17}$$

$$t_i^{\max} = \max\{t_i^k, \quad k = 1,\dots Q\} \tag{5.18}$$

Reverse processing function

Processing functions associated with a network target (output) transform targets into a better form for network training. Using the process described above, the range of the network targets are mapped to the range [-1,1]. Thus, in order for the trained network to give valuable results, its outputs should be reverse-processed using the same functions to produce output data with the same characteristics as the original user-provided targets. If the network output is a vector $\tilde{\boldsymbol{a}}$ with dimension S (same as the target), then the following reverse transformation should be done for every dimension i (i=1,...,S):

$$a_i = t_i^{\min} + \tilde{a}_i \cdot \left(t_i^{\max} - t_i^{\min} \right) \tag{5.19}$$

Using the above formula, the unnormalized network output $\boldsymbol{a}$ is in the same units as the original targets $\boldsymbol{t}$.

5.8 Training of NNs

The network parameters include all the weights and biases of the network. A neural network has to be configured such that the application of a set of inputs produces the desired set of outputs. Various methods to set the network parameters exist. One way would be to set the weights explicitly, using a priori knowledge. Another way is to 'train' the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule. A *training algorithm* or *learning rule* is defined as a procedure for modifying the network parameters in order to perform some particular task. Learning rules can be classified into two broad categories:

i. Supervised learning;

ii. Unsupervised learning.

5.8.1 Supervised learning

In *Supervised learning* or *Associative learning*, the training algorithm is provided with a set of examples (the *training set*), described in Eq. (5.12), of proper network behavior. These input-output pairs can be provided by an external teacher, or by the system which

contains the network (self-supervised). As the inputs are applied to the network, the network outputs are compared to the targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets.

5.8.2 Unsupervised learning

In *Unsupervised learning* or *Self-organization*, the weights and biases are modified in response to network inputs only, as there are no target outputs available. An output unit is trained to respond to clusters of pattern within the input. In this paradigm the system is supposed to discover statistically salient features of the input population. Unlike the supervised learning paradigm, in this case there is no a priori set of categories into which the patterns are to be classified, as the system must develop its own representation of the input stimuli. Most of unsupervised learning algorithms perform clustering operations. They categorize the input patterns into a finite number of classes. This is especially useful in such applications as vector quantization.

5.9 Back-Propagation Neural Network

From among many architectures of NNs, the most popular is the feed-forward multilayer NN, also called a multilayer perceptron (Haykin 1998) or *Back-Propagation Neural Network* (BPNN). Here, the output values are compared with the correct answer to compute the value of a predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has *learned* a certain target function. As the algorithm's name implies, the errors propagate backwards from the output nodes to the inner nodes. So technically speaking, back-propagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. In order to minimize the error, the derivative of the error function with respect to the network weights is calculated, and the weights are then changed such that the error decreases (thus going downhill on the surface of the error function). For this reason, back-propagation can only be applied on networks with differentiable activation functions. Often the term *back-propagation* is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in stochastic gradient descent. Back-propagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

A BPNN is a feed-forward, multilayer network of standard structure, i.e. neurons are not connected in the layer but they join the layer neuron with all the neurons of previous

and subsequent layers, respectively. A BPNN has a standard structure that can be written in short as

$$N - H_1 - H_2 - \ldots - H_{NL-1} - M \tag{5.20}$$

where N is the number of inputs, H_l is the number of neurons in the l-th hidden layer, NL is the number of layers (including the output layer) and M is the number of output neurons. Figure 5.18 depicts an example of a BPNN composed of an input layer with 4 neurons, two hidden layers with 3 neurons each and an output layer with 2 neurons, i.e. a 4-3-3-2 BPNN.

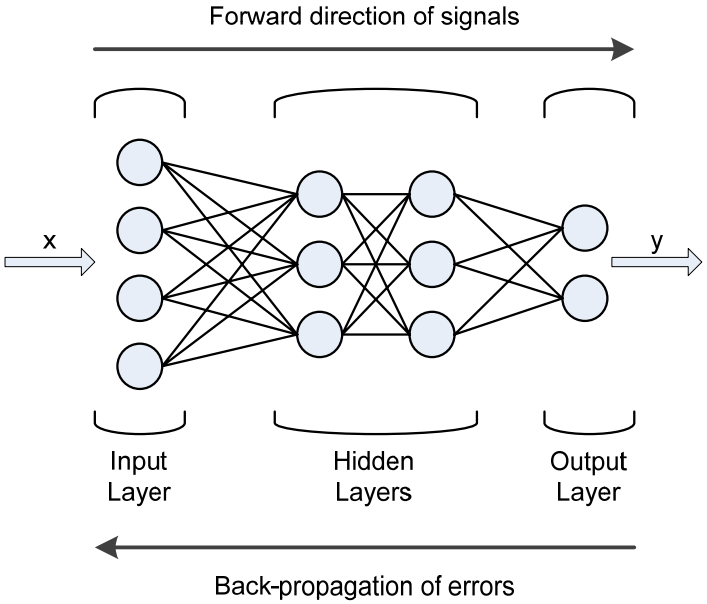


Figure 5.18 A three-layer 4-3-3-2 BPNN (input not counted as a layer).

The same 4-3-3-2 BPNN can be visualized using the abbreviated notation described in Section 5.5.5, as shown in Figure 5.19. The abbreviated notation for this network seems more complicated than the classical notation of Figure 5.18, yet it is quite useful for representing bigger networks with many neurons for each layer, where the classical notation with every neuron connected to every other for consecutive layers would be much more complicated.

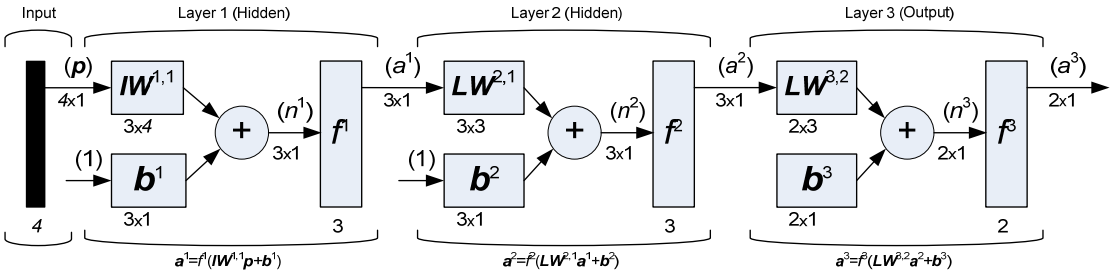


Figure 5.19 The abbreviated notation for the three-layer 4-3-3-2 BPNN.

The input layer serves only to introduce signals (values of input variables) and the hidden and output layers are composed of neurons (processing units) of the NN structure (Kuźniar and Waszczyzyn 2007). The values of weights $w^l_{i,j}$ and biases b^l_i are called network parameters. The number of network parameters *NNP* equals the total number of generalized weights (NN connections and neuron biases):

$$NNP = NNW + NNB = \left(N \cdot H_1 + \sum_{l=1}^{NL-1} (H_l \cdot H_{l+1}) \right) + \sum_{l=1}^{NL} H_l \tag{5.21}$$

where *NNW* is the number of weights and *NNB* is the number of biases. For instance, the network shown in Figure 5.18 has $NNW=4\cdot3+3\cdot3+3\cdot2=27$, $NNB=3+3+2=8$, thus the total number of NN parameters is $NNP=27+8=35$. The number of network parameters *NNP* can be used as a measure of the network size. For the NN design purposes it is of value to satisfy the inequality (well-posed formulation):

$$NNP \le L \cdot M \tag{5.22}$$

where *L* is the number of training patterns.

Back-propagation is a common supervised training algorithm. The term is an abbreviation for "backwards propagation of errors". It was first described in the PhD thesis of Paul Werbos (1974), but it wasn't until 1986, through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams (Rumelhart et al. 1986), that it gained recognition, and it led to a “renaissance” in the field of artificial neural network research (Hagan et al. 1996). Back-propagation is the generalization of the *Widrow-Hoff learning rule* (a.k.a. *Adaline rule*, *Delta rule*, *Least Mean Squares* (LMS) *Rule*) to multiple-layer networks and non-linear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way.

Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Standard in back-propagation is the gradient descent algorithm, in which the network weights are moved along the negative of the gradient of the performance function. The term *back-propagation* refers to the manner in which the gradient is computed for non-linear multilayer networks. There are a number of variations of the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods.

Properly trained back-propagation networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This generalization property makes it possible to train a network on a representative set of input/target pairs and get good results without training the network on all possible input/output pairs.

It is important to note that in order for the hidden layer of a BPNN to serve any useful function, multilayer networks must have non-linear activation functions for the multiple layers. A multilayer network using only linear activation functions is equivalent to a single layer linear network.

5.9.1 Summary of the back-propagation technique

The back-propagation technique follows the following steps:

1. Present a training sample to the NN.
2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a *scaling factor*, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat from *step 3* on the neurons at the previous level, using each one's "blame" as its error.

The values of weights $w^l_{i,j}$ and biases b^l_i can be put in order as components of the vector of generalized weights:

$$\boldsymbol{w} = \{ w_i \mid i = 1, \ldots, NNP \} \tag{5.23}$$

The computation of the network parameters is called training (learning) process. It is based on a training set of patterns $\mathcal{L}$, composed of known pairs of inputs and outputs (targets), i.e. input and output vectors $\boldsymbol{x}$, $\boldsymbol{t}$, respectively. After being trained, the network is tested on an independent testing set $\mathcal{T}$. The training and testing pattern sets can be written in the following form:

$$\mathcal{L} = \{ (x_i, t_i), \quad i = 1, \ldots, L \} \tag{5.24}$$

$$\mathcal{T} = \{ (x_j, t_j), \quad j = 1, \ldots, T \} \tag{5.25}$$

where L and T are the number of patterns in training and testing sets, respectively. The components of generalized weight vector w_i (network parameters) are iteratively computed by means of the following formula:

$$w_i(s+1) = w_i(s) + \Delta w_i(s) \tag{5.26}$$

where s is the number of iteration step. The learning formula for the weight increment can be written in the following form:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{5.27}$$

where η is the learning rate and E is the network error function (called the Least-Mean-Square error):

$$E = \frac{1}{2}\sum_{p=1}^{V}\sum_{i=1}^{M}\left(t_i^p - y_i^p \right)^2 = \frac{1}{2}\sum_{p}\sum_{i}(\delta_i^p)^2 \tag{5.28}$$

This formula is related to the output vectors $\boldsymbol{y}^p$ and the target vectors $\boldsymbol{t}^p$. The error $\delta^p{}_i$ corresponds to the i-th output for p-th pattern and $V=L, T$, the learning or the testing set.

Eq. (5.27) corresponds to the gradient method of steepest descent. This method is sensitive to values of the learning rate η. For small values of η the iteration process can be slowly convergent or even divergent below certain values of η_{crit}. Other learning methods can be also used, such as the *Resilient propagation* (Rprop) method.

Measures of errors

Besides the Least-Square-Error E defined in Eq. (5.28) there are also other error measures for evaluation of the accuracy of neural approximation. The most popular is the *Mean Square Error* (MSE):

$$MSE(V) = \frac{1}{V\,M}\sum_{p=1}^{V}\sum_{i=1}^{M}(\overline{t}_i^{(p)} - \overline{y}_i^{(p)})^2\,, \tag{5.29}$$

where $t_i^{(p)}, y_i^{(p)}$ are the target and neurally computed i-th outputs for p-th pattern. The values of $\overline{t}_i^{(p)}$ and $\overline{y}_i^{(p)}$ are usually scaled to the range [0.1, 0.9] if the sigmoid activation function is used. The formula of Eq. (5.29) can be applied to the computation of both the training error $MSE(L)$ and testing error $MSE(T)$ depending on the sets of patterns used.

5.9.2 Strengths and weaknesses of back-propagation learning

The strengths of the back-propagation learning technique include the following:

- It has great representation power;
- It has a wide practical applicability;
- It is easy to implement;
- It exhibits a good generalization power.

On the other hand, the weaknesses of the back-propagation learning technique are the following:

- Learning can sometimes take a long time to converge;

- The network can be essentially considered as a black box;
- The gradient descent approach can only guarantee a local minimum error;
- Generalization is not guaranteed even if the error is reduced to zero;
- There is no well-founded way to assess the quality of BP learning;
- Network paralysis may occur (learning is stopped, see Section 5.10.2);
- Selection of learning parameters can only be done by trial-and-error;
- BP learning is non-incremental. As a result, to include new training samples, the network must be re-trained with all old and new samples.

5.10 Problems with Neural Networks

5.10.1 Extrapolation

Compared to linear methods of function approximation, feed-forward Neural Networks exhibit great generality and flexibility, but also a significant drawback: they cannot extrapolate. By the term *extrapolation*, we denote the ability of an approximation method to give reliable results when confronted with data outside of the region used to calibrate the model. A neural network can map virtually any function by adjusting its parameters according to the presented training data. For unknown data points which fall into the region where the training data are present, interpolation can be done, giving reliable results, provided that he training has been done successfully. For regions of the variable space where no training data are available, extrapolation must be done and the output of the neural network cannot be considered reliable. Extrapolation remains a problem despite the fact that special types of NNs have been proposed for this reason (Dariani et al. 2007; Sari et al. 2007).

Figure 5.20 shows the performance of a NN trained to simulate the linear function $y=x$. The training data is within the interval [-20,20]. The NN is trained and then asked to provide results for the interval [-40,40]. It is clear that the NN performance within the range of the training data, where interpolation is done, is excellent, yet outside of the range of the training data, where extrapolation is done, performance is bad.

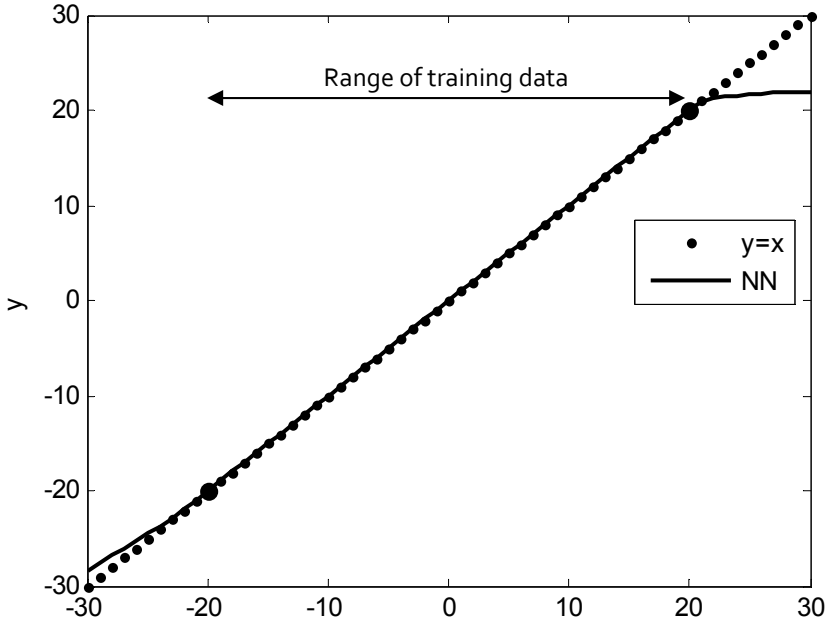


Figure 5.20 Performance of a NN trained to simulate the linear function *y=x*.

In order to avoid NN extrapolation, one should record the range of the variable space where training data is available. This can be done by calculating the convex hull of the training data set. The *convex hull* or *convex envelope* for a set of points $\mathcal{X}$ in a real vector space $\mathcal{V}$ is defined as the minimal convex set containing $\mathcal{X}$. In other words, it is the minimum volume polyhedron circumscribing the training points. If unknown data presented to the net are within this hull, the output of the net can be considered as reliable. The convex hull for a training data set with 50 patterns in a two-dimensional training space is shown in Figure 5.21.

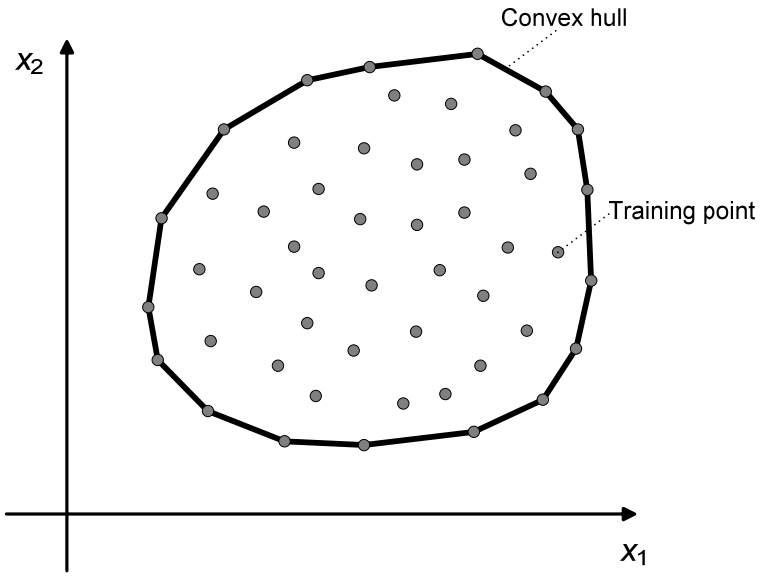


Figure 5.21 Training data in a two-dimensional space and the corresponding convex hull.

However, the concept of the convex hull is not always satisfactory since on the one hand the hull can be complicated to calculate and on the other hand, in case of non-convexities, training data may be absent even from regions inside the convex hull, as shown in Figure 5.22 for a training data set with 50 patterns in a two-dimensional training space.

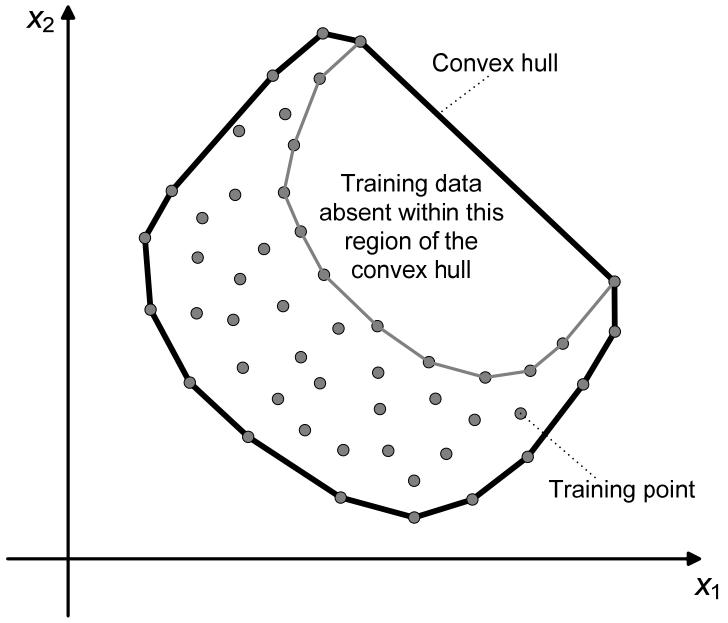


Figure 5.22 Non-convexity in the training data set, in a two-dimensional space.

In this case, for data points within the region of the convex hull where training data are missing, the performance of the NN can be poor, although interpolation is done instead of extrapolation.

A better method, proposed by Leonard et al. (1992), suitable for all types of networks, is to estimate the local density of training data by using Parzen windows (Parzen 1962). Radial basis function networks provide another elegant yet simple method of detecting extrapolation regions (Lohninger 1993). Shao et al. (1997) proposed a method for the calculation of confidence bounds for feed-forward NN models, taking into consideration the accuracy of the NN to predict the unknown data and the distribution of the available training data.

In any case, when using NNs to predict the structural response, it is advisable to select the training set in such a way that interpolation is always done by the NN, instead of extrapolation. This basic rule has been followed for all the NN applications of the thesis.

5.10.2 Network paralysis

Under some circumstances, the correction of the network parameters by NN training can cause no improvement to the output values. In this case, the training process can con-

tinue infinitely. This phenomenon, called *network paralysis*, can occur when a large ratio of the network neurons have big values of weights and therefore generate big values of input signals. Then also the output values take big values, which are at the boundaries of the sigmoid transfer function. At these positions, the derivative of the function approaches zero. This derivative is then used by the training algorithm in order to correct the network parameters, and therefore no correction is done. If this phenomenon happens to a big part of the network, then the learning process can in fact come to a halt.

It is difficult to determine when a network will face the problem of paralysis. Empirically it has been seen that rather small step sizes do not cause this problem, but on the other hand they can lead to very long training time. By normalizing properly the input values, as was shown in Section 5.7, the problem can be restricted.

5.10.3 Network over-training

The information contained in any data set of inputs and outputs for neural network training may not be fully accurate, as these data can contain also a small or bigger amount of "noise". The information contained can be generally classified into two groups: (i) major trends; and (ii) scatter (noise). In order for the network to maintain the precious property of generalization, it is desirable that the trained neural network learn the major trends in the data, but not the scatter. If the network also learns the scatter, then it is said to be *over-trained*. Over-trained neural networks lose their generalization capability. In order to avoid overtraining, the size of the network and the number of the training epochs should be in correspondence with the complexity of the problem, especially if the training data contains significant amount of noise. An excessive number of training epochs, or an excessive size of the neural network can cause network overtraining.

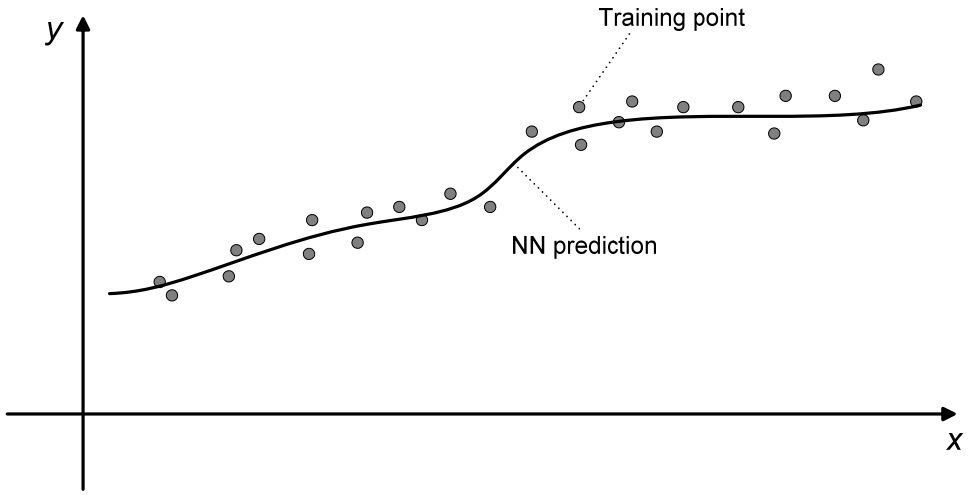


Figure 5.23 Training data points and NN prediction.

Consider a simple NN with one input and one output. Figure 5.23 shows the training data as points in the x-y plane. The data has obvious scatter: the major trend consists of two slopes and there is also a transition between them. Figure 5.23 shows also the prediction of a trained NN (bold line). The network has identified the two major trends and the transition and can generalize, giving valuable results.

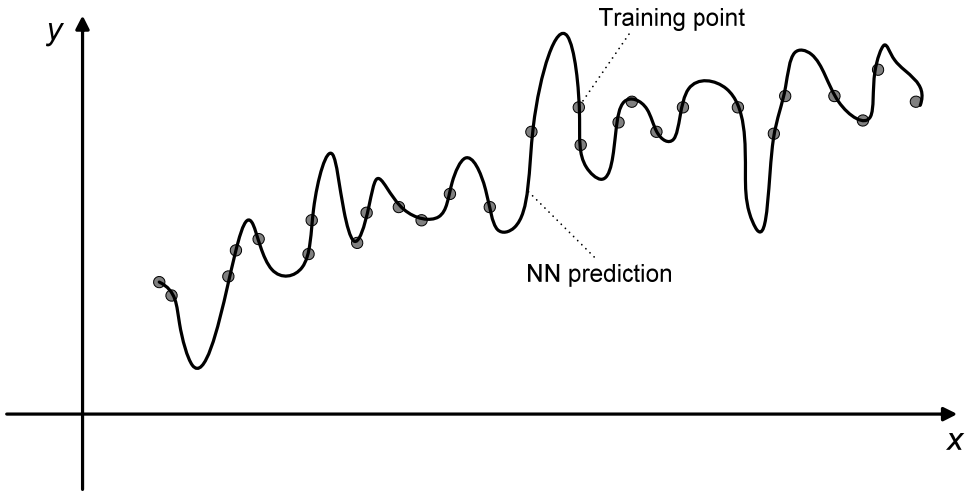


Figure 5.24 Non-convexity in the training data set, in a two-dimensional space.

Figure 5.24 shows the same training data set and the corresponding prediction of an *overtrained* NN (bold line). In this case, it is obvious that, although the approximation error for the training data itself is smaller than in the previous case, the approximation for other values of *x*, not included in the training set, will be in gross error. Overtraining resulted in the NN learning also the scatter, thus losing its generalization capability.

In order to avoid this problem, a validation data set can be used during the training process, as is the case in the present thesis. The validation data set is not taken into account for the training process, but it is used as an error measurement to stop the process. If, after a number of training epochs, the error for the training data still gets smaller, while at the same time the error for the validation data increases, then it is time to stop the training process in order to avoid network overtraining.

5.11 Neural Networks as metamodels in structural engineering

Over the last ten years Artificial Intelligence techniques have emerged as a powerful tool that could be used to replace time consuming procedures in many scientific or engineering applications. The principal advantage of a properly trained NN is that it requires a trivial computational effort to produce an approximate solution. Such approximations, if acceptable, appear to be valuable in situations where the actual response computations are intensive in terms of computing time and a quick estimation is required. For each problem a NN is trained utilizing information generated from a number of properly se-

lected analyses. The data from these analyses are processed in order to obtain the necessary input and output pairs, which are subsequently used to produce a trained NN. The training of a NN is an unconstrained minimization problem where the objective is to minimize the prediction error. In the case of structural optimization, the analysis corresponds to a finite element solution of the resulting equilibrium equations and the trained NN is then used to predict the response of the structure in terms of constraint function values due to different sets of design variables.

The use of artificial Neural Networks to predict finite element analysis outputs has been studied previously in the context of optimal design of structural systems (Adeli and Park 1995a; Adeli and Park 1995b; Arslan and Hajela 1997; Berke et al. 1993; Hajela and Berke 1991; McCorkle et al. 2003; Papadrakakis et al. 1998b; Shieh 1994) and also in some other areas of structural engineering applications, such as structural damage assessment (Zacharias et al. 2004), structural reliability analysis (Hurtado and Alvarez 2001; Nie and Ellingwood 2004; Papadrakakis et al. 1996a), finite element mesh generation or fracture mechanics (Gunaratnam and Gero 1994; Khan et al. 1993; Papadrakakis et al. 1996a; Stephens and VanLuchene 1994; Theocaris and Panagiotopoulos 1993; Topping and Bahreininejad 1997), evaluation of buckling loads of cylindrical shells with geometrical imperfections (Waszczyszyn and Bartczak 2002) and others.

NNs have been recently applied to the solution of the equilibrium equations resulting from the application of the finite element method in connection to reanalysis type of problems, where a large number of finite element analyses is required. Reanalysis type of problems are encountered, among others, in the reliability analysis of structural systems using *Monte Carlo Simulation* (MCS) and in structural optimization using *Evolutionary Algorithms* (EAs). In these problems NNs have been proved to give very satisfactory results (Papadrakakis et al. 1998b; Papadrakakis et al. 1996a).

In the present thesis, the Neural Networks have been used in order to assist the methodologies for the problems of optimum structural design considering uncertainties, namely the *Reliability-Based Design Optimization* (RBDO), the *Robust Design Optimization* (RDO) and the *Reliability-based Robust Design Optimization* (RRDO) methodologies, by providing computationally inexpensive estimates of the structural response or the constraints checks, as will be described in detail in Chapter 6 (Design Optimization Considering Uncertainties), as well as Chapter 8 (Numerical Applications – Probabilistic Optimization).

Chapter 6

6 Design Optimization Considering Uncertainties

6.1 The concept of probabilistic design optimization

The primal engineering objective during the design of any structural system is the minimization of its construction cost, the improvement of its structural performance and at the end the minimization of its total life-cycle cost. Improvements during the design stage can be achieved either by simply using design rules based on experience and trial-and-error, or by an automated way using structural optimization procedures, described in detail in Chapters 3 and 4. Taking into account the complexity of a structural optimization problem, it is obvious that finding the global optimum solution is not an easy task. In real-world conditions the significance of any "optimum" solution would be limited if the uncertainties involved in the geometric and material description of the structure as well as the loading conditions are not taken into account. Real-world structures have always imperfections which induce deviations from the nominal state assumed by the design codes.

In *Deterministic Design Optimization* (DDO), where the formulation of the optimization problem ignores scatter of any kind of the parameters affecting its response, the designs are often driven to the limit of the design constraints, leaving little space for uncertainties. Consequently, a deterministically optimum design may result in an infeasible design due to unavoidable scattering of the values of its basic parameters affecting its structural response. Additionally, the deterministic optimum is never materialized in an absolute way and as a result a near optimal solution is always applied in practice, which can be associated with an undesirable high probability of failure, due to the influence of the uncertainties present during the modeling and manufacturing phases and in the external operating conditions. Uncertainties are inherently present and need to be accounted for in the design optimization process as they may lead to large variations in the performance characteristics of the system and to a high probability of failure. Optimized deterministic designs obtained without considering uncertainties can be unreliable and might lead to catastrophic structural failure.

In order to account for the randomness of the parameters that affect the response of the structure, a different formulation of the optimization problem has to be introduced, based on probabilistic treatment of the uncertainties involved. The development of probabilistic analysis methods over the last two decades has stimulated the interest for considering randomness and uncertainty in the formulation of structural optimization problems (Schuëller 2005; Tsompanakis et al. 2008).

Probabilistic-based design optimization methodologies can be generally classified in the following two formulations:

i. *Reliability-Based Design Optimization* (RBDO) (Acar and Haftka 2007; Agarwal and Renaud 2004; Du et al. 2006; Frangopol 1995; Frangopol and Maute 2003; Gasser and Schuëller 1997; Gunawan and Papalambros 2006; Jensen et al. 2008; Jiang et al. 2000; Lagaros et al. 2008a; Liang et al. 2007; Moses 1997; Nikolaidis 2007; Pu et al. 1997; Rackwitz 2004; Repalle and Grandhi 2005; Rosenblueth and Mendoza 1971; Royset et al. 2001; Sorensen and Tarp-Johansen 2005; Streicher and Rackwitz 2004);

ii. *Robust Design Optimization* (RDO) (Anthony and Keane 2003; Beyer and Sendhoff 2007; Doltsinis and Kang 2004; Doltsinis et al. 2005; Lagaros and Papadrakakis 2007; McAllister and Simpson 2003; Park et al. 2006; Parkinson et al. 2007; Rangavajhala et al. 2007; Ray and Smith 2006; Schumacher and Olschinka 2008; Zang et al. 2005).

A reliable design is one for which the probability of failure or the probability of exceeding certain limits of the system is low. The main goal of RBDO methods is to find the optimum design, which at the same time satisfies the objective of the minimum weight in conjunction with limitations on the allowable probability of failure or probability of exceedance of certain characteristic structural response quantities.

On the other hand, a robust design is one for which the variation of the performance function is minimal. RDO methods primarily seek to minimize the influence of random variations of the nominal structural dimensions, material parameters and loading on the response of the structure, aiming at a reduction of the spread of critical responses.

Designing for both reliability and robustness is extremely desirable in structural engineering, as will be described in detail in the following sections.

6.2 Reliability-Based Design Optimization (RBDO)

6.2.1 Introduction

In deterministic optimization, the target is to reduce the initial structural cost under a number of constraints. In this type of problem, the structural safety is assumed to be ensured by the introduction of the safety factors within the constraints. It is assumed that the safety factors are appropriate whatever the optimal configuration. Yet, for most sys-

tems it can be shown that the safety level is not independent of the selected optimal design parameters.

Uncertainties in the loading conditions, material properties and analysis modeling contribute to making the performance of the optimal design different from the expected one. In this sense, the optimization process has a large effect on the structural reliability. The simplistic safety factor approach cannot ensure the required level of reliability and safety, as it does not explicitly consider the probability of failure regarding some performance criteria. In other words, the optimal design obtained from deterministic optimization procedures does not necessarily meet the reliability targets.

Reliability theory is introduced in structural engineering and structural optimization in order to consider, in a more rational way, all existing uncertainties which are not known with the desirable degree of precision and can influence structural response. Reliability is recognized as a safety constraint in structural engineering, therefore any optimum design under reliability constraints should balance both cost and safety. While in DDO, stress and displacement constraints are considered in accordance with the design code safety factors, in RBDO additional probabilistic constraints are incorporated into the optimization procedure leading to unbiased estimates of the structural performance and, subsequently, to the determination of designs that are located within a range of target failure probabilities. The probabilistic constraints enforce the condition that the probability violation is smaller than a certain threshold value.

RBDO aims at finding the optimal solution that fulfills the prescribed reliability requirements, characterized by a low probability of failure or in other words the one that assures that the failure limit state is kept sufficiently far from the operating point. The obtained failure surface must lie on the iso-reliability level corresponding to the prescribed safety target (Chateauneuf 2008). The solution can be quite different from the deterministic optimization where homothetic reduction of the design space is applied. In this sense, RBDO can really ensure optimal cost, without compromising the structural safety. In RBDO problems, there is a trade-off between obtaining higher reliability and lowering cost. RBDO allows us to consider the safety margin evolution, leading to the settlement of the best compromise between the cost and the required reliability. The task is complicated, due to the inherent non-deterministic nature of the input information.

The probabilistic framework allows for a consistent treatment of both cost and safety. The RBDO problem can be also considered as a multi-objective optimization problem where the objective is to minimize the cost and maximize the structural safety (Kuschel and Rackwitz 1997; Kuschel and Rackwitz 2000). It is generally acceptable that reliability and economy have conflicting requirements that must be considered simultaneously in the optimization process. The most common RBDO formulations try to deal with the problem indirectly, either by combining the two objectives into one weighted objective, or more often by transforming the safety objective into a probabilistic constraint. A more accurate formulation would be to deal with the multi-objective problem directly, so that

the designer can obtain the Pareto Front curve and decide between Pareto optimal configurations in order to make consistent choices in the design process.

The first step in RBDO is to characterize the important uncertain variables and the failure modes. In most engineering applications, the uncertainty is generally characterized using probability theory. The probability distributions of the random variables are obtained using statistical models. In designing models with multiple failure modes, it is important that the design be sufficiently reliable with respect to each of the critical failure modes or to the overall system failure. In a RBDO formulation, the critical failure modes in deterministic optimization are replaced with constraints on probabilities of failure corresponding to each of the failure driven modes or with a single constraint on the system probability of failure.

RBDO aims at searching for the best compromise between cost reduction and reliability assurance, by considering system uncertainties. RBDO can produce a more efficient design than a deterministic approach without sacrificing safety, or alternatively, it can yield a safer design than a deterministic approach for a given maximum allowable cost. The solution of a RBDO problem is not easy, even for simple structures, let alone realistic complicated structural systems with many design variables and/or random variables. The difficulty lies in the consideration of the reliability constraints which require a large computational effort. While the optimization process is carried out in the space of the design variables, the reliability analysis is performed in the space of random variables, where a lot of numerical calculations are required to evaluate the failure probability.

6.2.2 Formulation of a RBDO problem

The formulation of a RBDO problem can be written as follows:

$$
\begin{aligned}
&\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}), \quad \boldsymbol{x} = [x_1,\ldots,x_n]^{\mathrm{T}}, \quad f \in \mathbb{R} \\
&\text{Subject to} \\
&\boldsymbol{g}(\boldsymbol{x}) \le 0, \quad \boldsymbol{g} \in \mathbb{R}^p \\
&\boldsymbol{h}(\boldsymbol{x}) = 0, \quad \boldsymbol{h} \in \mathbb{R}^q \\
&p\left(y(\boldsymbol{x},\underline{\boldsymbol{r}}) < 0 \right) \le p_{\mathrm{f}}, \quad \underline{\boldsymbol{r}} \in \mathbb{R}^m \\
&x_i \in X_i \quad \text{for } i = 1,\ldots,n
\end{aligned}
\tag{6.1}
$$

where:

- $\boldsymbol{x} = [x_1,\ldots,x_n]^{\mathrm{T}}$ is the vector of the n design variables.
- $\underline{\boldsymbol{r}} = [\underline{r}_1,\ldots,\underline{r}_m]^{\mathrm{T}}$ is the vector of the m random variables.
- X_i is the set of x_i, which may be continuous, discrete or integer. The whole design space for the n design variables can be denoted as $\mathcal{X}$.
- $\boldsymbol{f}(\boldsymbol{x})$ is the scalar function to be minimized.

- $\boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} = [g_1(\boldsymbol{x}), \ldots, g_p(\boldsymbol{x})]$ is the vector function of p inequality constraints.
- $\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}} = [h_1(\boldsymbol{x}), \ldots, h_q(\boldsymbol{x})]$ is the vector function of q equality constraints.
- $p(\cdot)$ denotes the probability of a random event.
- $y(\boldsymbol{x}, \underline{\boldsymbol{r}})$ is the limit state function, thus $p(y(\boldsymbol{x}, \underline{\boldsymbol{r}}))$ denotes the probability of failure of the design.
- p_f is the threshold value for the probability of failure.

Using the above formulation, the structure is designed so that the probability of failure does not exceed a certain threshold value p_f during a prescribed period (life cycle).

It should be noted that in the above formulation, a certain structural parameter, such as the size of a structural cross section, can belong to both the design variables and the random variables of the problem. In this case, the mean value of the specific parameter will play the role of the design variable for the calculation of the objective function and the deterministic constraints, while the *Probability Density Function* (PDF) of the parameter will be taken into account for the calculation of the probability of failure using a stochastic analysis process.

6.3 Robust Design Optimization (RDO)

6.3.1 The concept of robustness

The requirement of practical, sometimes simplifying approaches can be considered as the implicit but dominating rule in engineering design. This has fostered the concept of robustness, meaning a product that exhibits strength with respect to variations or fluctuations of random, uncontrollable or unknown parameters. A robust product assures the engineer that it can absorb such fluctuation without compromising its quality, which is its main feature. In general, robust design orientation aims at overcoming the need of considering particular uncertain situations and guaranteeing the imperturbability of the system under the presence of unknown, unpredictable or random parameters. However, the question for a quantitative measure of the uncertainty in extreme situations is not addressed directly by the robustness approach.

A robust solution is defined as one which is less sensitive to the perturbation of the decision variables in its neighborhood. Consider a single-objective optimization problem with one design variable, where the objective function to be minimized is shown in Figure 6.1. In a deterministic formulation of the optimization problem, the global optimum is represented by point *A*, while solution *B* is only a local optimum.

On the other hand, it is clear that solution *A* is quite sensitive to the variable perturbation and often cannot be recommended in practice, despite having a better objective function value than solution *B*. Of the two optimal solutions, solution *B* is considered

robust as a small variation in the design variable does not alter the objective function value of the solution significantly.

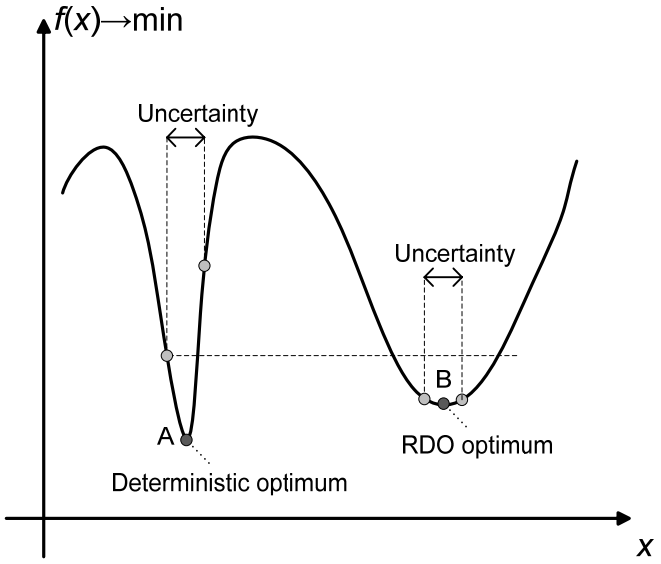


Figure 6.1 Illustration of deterministic versus robust solutions in a scalar optimization problem with one design variable.

The concept of robustness discussed above for single-objective optimization can be extended also to multi-objective optimization cases. In Figure 6.2 two Pareto-optimal solutions (*A* and *B*) are checked for their sensitivity in the design variable space, at the left of the figure. The corresponding solutions are also shown in the objective function space at the right of the figure. Since a local perturbation of point *A* causes a large change in objective values, this solution may not be considered as a robust solution, whereas solution *B* which does not cause a large change in objective values due to a local perturbation in its vicinity, can be considered as a robust solution. Of course, although the two solutions are both Pareto optimal, they are not directly comparable as they correspond to different decision criteria in terms of the objective function values f_1 and f_2.

To qualify as a robust solution, each Pareto-optimal solution has to demonstrate its insensitivity towards small perturbations in its decision variable values. In multi-objective optimization problems the sensitivity has to be established with respect to all *m* objectives. That is, a combined effect of variations in all *m* objectives has to be used as a measure of sensitivity to variable perturbation, while there are many solutions to be checked for robustness.

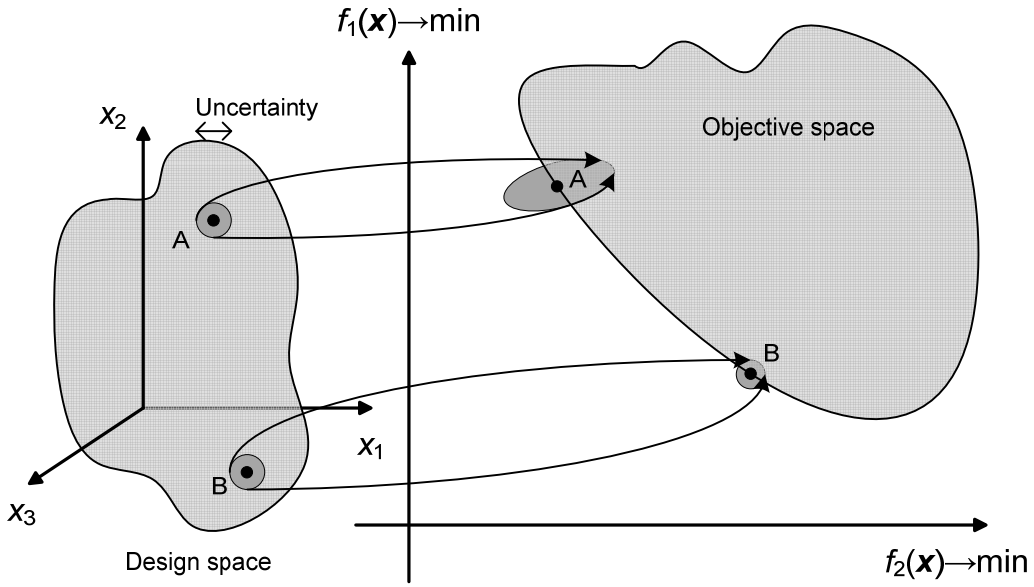


Figure 6.2 Illustration of deterministic versus robust solutions in a multi-objective optimization problem with two objective functions.

Although a great deal of studies has been done during the last three decades in structural optimization, those devoted to RDO are rather limited. Chen and Lewis (1999) presented some preliminary results of robust design for multidisciplinary optimization applying efficient methods for the uncertainty analysis. Lee and Park (2001) solved a RDO problem where the multi-objective problem considered had two criteria to be minimized, the mean value and the standard deviation of the structural weight. The multi-objective problem was solved using the weighting sum method in the context of a mathematical programming algorithm. Sandgren and Cameron (2002) proposed a hybrid genetic/non-linear programming algorithm for the solution of the multi-objective problem in the framework of RDO. Messac and Ismail-Yahaya (2002) developed the flexible physical programming-based RDO methodology that formulates the RDO problem in terms of physically meaningful design performance degradation levels. Jung and Lee (2002) incorporated the probability of feasibility into the RDO problem, where each probability constraint was transformed into a sub-optimization problem by the advanced first-order second moment method. Doltsinis and Kang (2004) dealt with the RDO problem considering the minimization of the mean value and the standard deviation of a nodal displacement and treating the structural weight as a constraint function.

6.3.2 Formulation of a RDO problem as a Multi-objective Optimization Problem

The formulation of a Robust Design Optimization problem can be written as a multi-objective optimization problem as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \left[f(\boldsymbol{x}), \sigma_u(\boldsymbol{x}, \underline{\boldsymbol{r}}) \right]^{\mathrm{T}}, \quad f \in \mathbb{R}, \sigma_u \in \mathbb{R}$$

Subject to

$$\boldsymbol{g}(\boldsymbol{x}) \leq 0, \quad \boldsymbol{g} \in \mathbb{R}^p \tag{6.2}$$

$$\boldsymbol{h}(\boldsymbol{x}) = 0, \quad \boldsymbol{h} \in \mathbb{R}^q$$

$$x_i \in X_i \quad \text{for } i = 1, \ldots, n$$

where:

- $\boldsymbol{x} = [x_1, \ldots, x_n]^{\mathrm{T}}$ is the vector of the n design variables.
- $\underline{\boldsymbol{r}} = [\underline{r}_1, \ldots, \underline{r}_m]^{\mathrm{T}}$ is the vector of the m random variables.
- X_i is the set of x_i, which may be continuous, discrete or integer. The whole design space for the n design variables can be denoted as $\mathcal{X}$.
- $f(\boldsymbol{x})$ is the scalar function to be minimized, usually the weight of the structure.
- $\sigma_u(\boldsymbol{x},\underline{\boldsymbol{r}})$ is a scalar function (a statistical quantity), the standard deviation of a response measure of the structure to be minimized. This response measure is in most cases associated with displacements, i.e. the maximum displacement of the structure.
- $\boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} = [g_1(x), \ldots, g_p(x)]$ is the vector function of p inequality constraints.
- $\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}} = [h_1(x), \ldots, h_q(x)]$ is the vector function of q equality constraints.

6.4 Relationship between RBDO and RDO formulations

RDO methods primarily seek to reduce the spread of critical responses, while RBDO methods seek to control the probability of failure. Since the reduction of the response spread not necessarily precludes a failure with regard to extreme cases, the two methods can be considered as complementary. In general, it is desirable to have available methods that yield a design that satisfies both reliability and robustness criteria. The nature of these two alternative methods can be explained with the help of Figure 6.3, which shows two alternative probability density functions of the structural response. RDO aims to reduce the spread, while RBDO is intended to limit the probability of surpassing a critical threshold value (dashed line) (Hurtado 2008).

It should be noted that the effect pursued by RBDO can be indirectly obtained by applying RDO, due to the fact that in this case the reduction of the spread implies a reduction of the failure probability. The reliability refers to the occurrence of extreme events, whereas robustness refers to the spread of the structural response under a large variation of the input parameters. This is assumed to assure a narrow response density function, which in turn assures most of the times a low failure probability. However, to a significant spread of the structural response may correspond a low failure probability (dotted line) because the definition of the limit state can be such that the possibility of surpassing it is very rare, as the situation it describes is rather extreme.

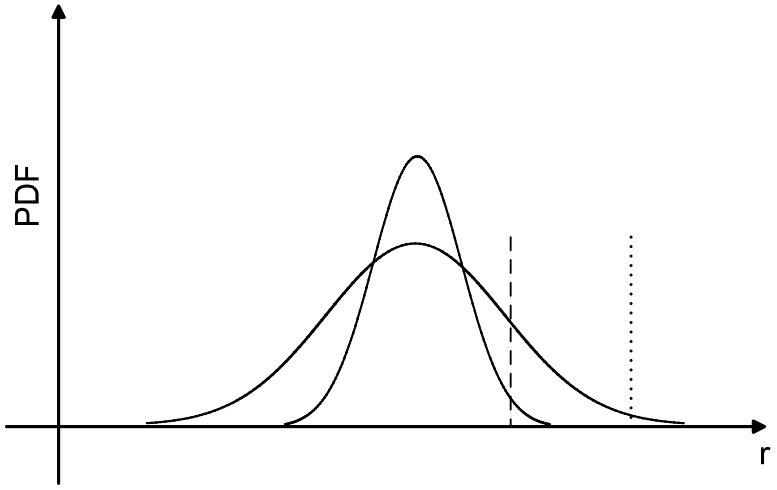


Figure 6.3 Robust and Reliability-based design options.

The comparison between RDO and RBDO cannot be conclusive because, as it is shown in Figure 6.3, the choice depends on the critical thresholds selected for reliability estimations. Besides, the relationship between statistical moments and probabilities is highly non-linear.

For the above reasons, both approaches are valuable and complementary for taking into account uncertainties in structural design. Since both kinds of designs correspond to a different way of incorporating the uncertainties and to different goals, a method allowing monitoring both the moments (mean and variance) and the failure probabilities, with a low computational cost, would be most suitable.

6.5 Reliability-based structural optimization using metamodel assisted ES

In the reliability analysis of structures considering elasto-plastic material behavior using MCS, the computed critical load factors are compared to the corresponding external loading leading to the computation of the probability of structural failure. The probabilistic constraints enforce the condition that the probability of a local failure of the system or the global system failure is smaller than a certain value (i.e. 10^{-5} to 10^{-3}). In the present thesis, the overall probability of failure of the structure, as a result of a limit state elasto-plastic analysis, is taken as the global reliability constraint.

Three metamodel-assisted RBDO methodologies are investigated:

i. RBDO-NN1 methodology: NN used for the deterministic and probabilistic constraints check.

ii. RBDO-NN2 methodology: NN prediction of the maximum load capacity.

iii. RBDO-NN3 methodology: A two-level NN for RBDO, combining the other two methodologies.

6.5.1 RBDO-NN1 methodology: NN used for the deterministic and probabilistic constraints check

In this methodology, a trained NN that utilizes information generated from a number of properly selected design vectors is used to perform both the deterministic and probabilistic constraints checks that are needed during the optimization process. After the selection of a suitable NN architecture, the training procedure is performed using a number (M) of data sets in order to obtain the input/output pairs needed for the NN training. The trained NN is then applied to predict the response of the structure in terms of deterministic and probabilistic constraints checks due to different sets of design variables.

The combined RBDO-NN1 optimization procedure is performed in two phases. The first phase includes the training set selection, the corresponding structural analysis and MCS for each training set required to obtain the necessary input/output data for the NN training, and finally the training and testing of a suitable NN configuration. The second phase is the ES optimization stage where the trained NN is used to predict the response of the structure in terms of the deterministic and probabilistic constraints checks due to different sets of design variables. The RBDO-NN1 implementation is described in Figure 6.4.

NN training phase:

1. *Training set selection step*: Select M input patterns.
2. *Deterministic constraints check*: Perform the check for each input pattern vector.
3. *Monte Carlo Simulation step*: Perform MCS for each input pattern vector.
4. *Probabilistic constraints check*: Perform the check for each input pattern vector.
5. *Training step*: Training of the NN.
6. *Testing step*: Test the trained NN.

ES-NN optimization phase:

1. Parents Initialization.
2. *NN (Deterministic-Probabilistic) constraints check*: All parents become feasible.
3. Offspring generation.
4. *NN (Deterministic-Probabilistic) constraints check*: If satisfied continue, else go to step 3.
5. Parents' selection step.
6. Convergence check.

Figure 6.4 The RBDO-NN1 methodology:
NN used for the deterministic and probabilistic constraints check.

6.5.2 RBDO-NN2 methodology: NN prediction of the maximum load capacity

In this methodology the limit state elasto-plastic analyses required during the MCS are replaced by the NN prediction of the structural behavior up to collapse. For every MCS a NN is trained utilizing available information generated from selected conventional elasto-plastic analyses. The limit state analysis data is processed to obtain input and output pairs, that are used for the NN training. The trained NN is then used to predict the critical load factor due to different sets of basic random variables.

1. Parents Initialization.
2. Deterministic constraints check: all parents become feasible.
3. Monte Carlo Simulation step:
   a. Selection of the NN training set.
   b. NN training for the limit load.
   c. NN testing.
   d. Perform MCS using NN.
4. Probabilistic constraints check: all parents become feasible.
5. Offspring generation.
6. Deterministic constraints check: if satisfied continue, else go to step 5.
7. Monte Carlo Simulation step:
   a. Selection of the NN training set.
   b. NN training for the limit load.
   c. NN testing.
   d. Perform MCS using NN.
8. Probabilistic constraints check: if satisfied continue, else go to step 5.
9. Parents' selection step.
10. Convergence check.

Figure 6.5 The RBDO-NN2 methodology:
NN prediction of the maximum load capacity.

At each ES cycle (generation) a number of MCS is carried out. In order to replace the time consuming limit state elasto-plastic analyses by predicted results obtained with a trained NN, a training procedure is performed based on the data collected from a number of conventional limit state elasto-plastic analyses. After the training phase is concluded the trained NN predictions replace the conventional limit state elasto-plastic ana-

lyses, for the current design. For the selection of the suitable training pairs, the sample space for each random variable is divided into equally spaced distances. The central points within the intervals are used as inputs for the limit state analyses. The RBDO-NN2 implementation is described in Figure 6.5.

In reliability analysis of elastoplastic structures using MCS, the computed critical load factors are compared to the corresponding external loading leading to the computation of the probability of structural failure according to Eq. (2.45). By approximating the "exact" solution with a NN prediction of the critical load factor, the accuracy of the predicted p_f depends not only on the accuracy of the NN prediction of the critical load factor but also on the sensitivity of p_f with regard to a slightly modified, due to the NN approximation, sample space of resistances. This sensitivity is represented in Figure 6.6 by the ratio of the shaded area over the total area defined by the probability density distributions and the failure function on the unsafe side. It occurs that the error due to this sensitivity is always present but is more pronounced in low probability estimations where the shadowed area becomes significant compared to the total area that defines the low probability of failure. Thus, the use of *Importance Sampling* (IS) technique, described in detail in Section 2.9, is expected to be beneficial since the sampling is performed in an area of high probabilities. In this case, as the ratio of the shaded area over the total area is decreased, the introduced error can be substantially reduced.

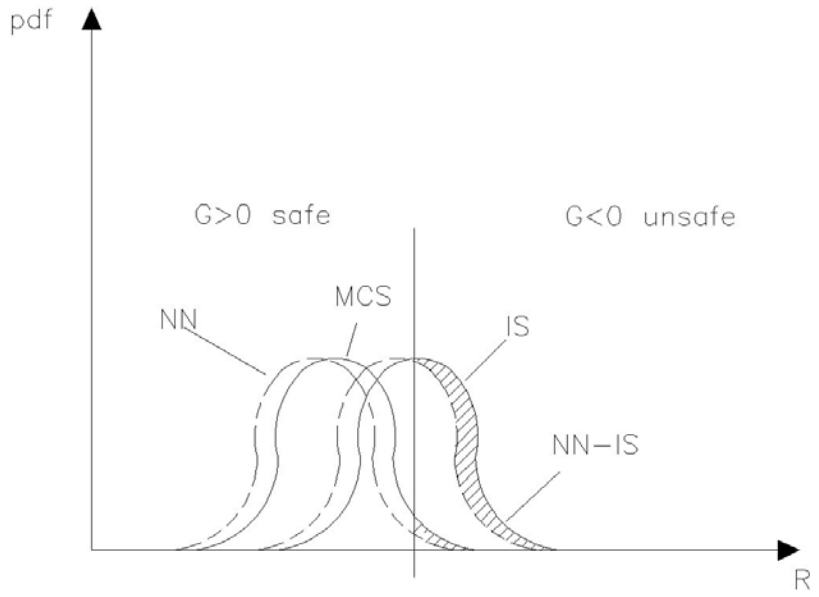


Figure 6.6 Sensitivity of p_f prediction to different sample space of resistances.

6.5.3 RBDO-NN3 methodology: A two-level NN for RBDO, combining the other two methodologies

This methodology combines the two other methodologies as follows: a trained NN1 following the first metamodel-assisted RBDO methodology RBDO-NN1, utilizes informa-

tion generated from a number of properly selected design vectors in order to perform both the deterministic and probabilistic constraints checks that are needed during the optimization process. The difference in comparison to the first RBDO-NN1 methodology is that a second NN2 is used, similarly to the way that the second metamodel-assisted RBDO methodology RBDO-NN2 employs the NN, in order to assist the training of the NN1. The RBDO-NN3 implementation is described in Figure 6.7.

NN1 training phase:

1. *Training set selection step*: Select *M* input patterns.
2. *Deterministic constraints check*: Perform the check for each input pattern vector.
3. *Monte Carlo Simulation step*:
   a. Selection of the NN2 training set.
   b. NN2 training for the limit load.
   c. NN2 testing.
   d. Perform MCS using NN2.
4. *Probabilistic constraints check*: Perform the check for each input pattern vector.
5. *Training step*: Training of the NN1.
6. *Testing step*: Test the trained NN1.

ES-NN optimization phase with NN1:

1. Parents Initialization.
2. *NN1 (Deterministic-Probabilistic) constraints check*: All parents become feasible.
3. Offspring generation.
4. *NN1 (Deterministic-Probabilistic) constraints check*: If satisfied continue, else go to step 3.
5. Parents' selection step.
6. Convergence check.

Figure 6.7 The RBDO-NN3 methodology:
A two-level NN for RBDO, combining the other two methodologies.

The combined RBDO-NN3 optimization procedure is performed in two phases:

i. The first phase includes the training set selection, the corresponding structural analysis and MCS (performed with the help of NN2) for each training set required to obtain the necessary input/output data for the NN1 training, and finally the training and testing of a suitable NN1 configuration;

ii. The second phase is the ES optimization stage where the trained NN1 is used to predict the response of the structure in terms of the deterministic and probabilistic constraints checks due to different sets of design variables.

6.6 Reliability-based Robust Design Optimization (RRDO)

The *Reliability-based Robust Design Optimization* (RRDO) problem, first addressed by Youn and Choi (2004; Youn et al. 2007), is also implemented in the present thesis in order to account for the influence of the probabilistic constraints in the framework of RDO of real-world structures. The key difference between the RDO and RRDO formulations lies in the probabilistic constraints that are taken into account in the RRDO formulation, as opposed to a RDO formulation where all constraints are deterministic.

6.6.1 Formulation of a RRDO problem as a Multi-objective Optimization Problem

A structural combined Reliability-based Robust Design Optimization problem can be formulated as a two objective optimization problem where an additional, to the initial construction cost, objective function is considered, related to the influence of the random nature of structural parameters and loading conditions on the performance of the structure, while probabilistic constraints are also taken into account. The aim is to minimize both the initial construction cost and the variance of the response of the structure. In the RRDO formulation, the probability of failure of the structure or the probability of violation of the constraints is taken into account as an additional set of constraints together with the deterministic constraints imposed by the RDO formulation.

Thus, the mathematical description of the generic RRDO problem implemented in this thesis is given as follows

$$
\begin{aligned}
&\min_{\boldsymbol{x}\in\mathbb{R}^n} \left[f(\boldsymbol{x}), \sigma_u(\boldsymbol{x}, \underline{\boldsymbol{r}}) \right]^{\mathrm{T}}, \quad f \in \mathbb{R}, \sigma_u \in \mathbb{R} \\
&\text{Subject to} \\
&\boldsymbol{g}(\boldsymbol{x}) \le 0, \quad \boldsymbol{g} \in \mathbb{R}^p \\
&\boldsymbol{h}(\boldsymbol{x}) = 0, \quad \boldsymbol{h} \in \mathbb{R}^q \\
&p\left(y(\boldsymbol{x}, \underline{\boldsymbol{r}}) < 0 \right) \le p_{\mathrm{f}}, \quad \underline{\boldsymbol{r}} \in \mathbb{R}^m \\
&x_i \in X_i \quad \text{for } i = 1, \ldots, n
\end{aligned}
\tag{6.3}
$$

where:

- $\boldsymbol{x} = [x_1, \ldots, x_n]^{\mathrm{T}}$ is the vector of the n design variables.
- $\underline{\boldsymbol{r}} = [\underline{r}_1, \ldots, \underline{r}_m]^{\mathrm{T}}$ is the vector of the m random variables.
- X_i is the set of x_i, which may be continuous, discrete or integer. The whole design space for the n design variables can be denoted as X.
- $f(\boldsymbol{x})$ is the scalar function to be minimized.

- $\sigma_u(\boldsymbol{x},\underline{\boldsymbol{r}})$ is a scalar function (a statistical quantity), the standard deviation of a response measure of the structure to be minimized. This response measure is in most cases associated with displacements, i.e. the maximum displacement of the structure.
- $\boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} = [g_1(x), \ldots, g_p(x)]$ is the vector function of p inequality constraints.
- $\boldsymbol{h}(\boldsymbol{x})^{\mathrm{T}} = [h_1(x), \ldots, h_q(x)]$ is the vector function of q equality constraints.
- $p(\cdot)$ denotes the probability of a random event.
- $y(\boldsymbol{x},\underline{\boldsymbol{r}})$ is the limit state function, thus $p(y(\boldsymbol{x}, \underline{\boldsymbol{r}}))$ denotes the probability of failure of the design.
- p_f is the threshold value for the probability of failure.

6.7 RRDO probabilistic analysis based on NN predictions

Despite the improvements achieved on the efficiency of the computational methods for treating reliability analysis problems, they still require disproportional computational effort for practical reliability analysis problems. In probabilistic analysis of structures, the Monte Carlo Simulation method, discussed in detail in Section 2.6, is particularly suitable when an analytical solution is not attainable. Despite the improvements achieved by the Latin Hypercube Sampling technique and other variance reduction techniques, described in detail in Section 2.7, MCS still requires disproportional computational effort for reliability analysis of realistic problems with a large number of random variables. This is the reason why very few successful numerical investigations are known in estimating the probability of failure and are mainly restricted to simple elastic frames and trusses (Jiang et al. 2007).

The formulation of the RRDO problem of Eq. (6.3) requires the implementation of reliability analysis in order to calculate $\sigma_u(\boldsymbol{x},\underline{\boldsymbol{r}})$ and $p_{v,\max}(\boldsymbol{x},\underline{\boldsymbol{r}})$. The computational difficulties described in the previous paragraph are also present in the implementation of the RRDO problem. In order to reduce significantly the computational cost of the RRDO procedure, a Neural Network based methodology for RRDO is also proposed in the present thesis.

6.7.1 NN-based MCS probabilistic analysis for RRDO

The methodology takes advantage of the global approximation capabilities of Neural Networks. In the methodology, shown schematically with the flowchart of Figure 6.8, for every new candidate Pareto optimum design, a new NN is trained to replace the finite element analyses required during the Monte Carlo Simulations.

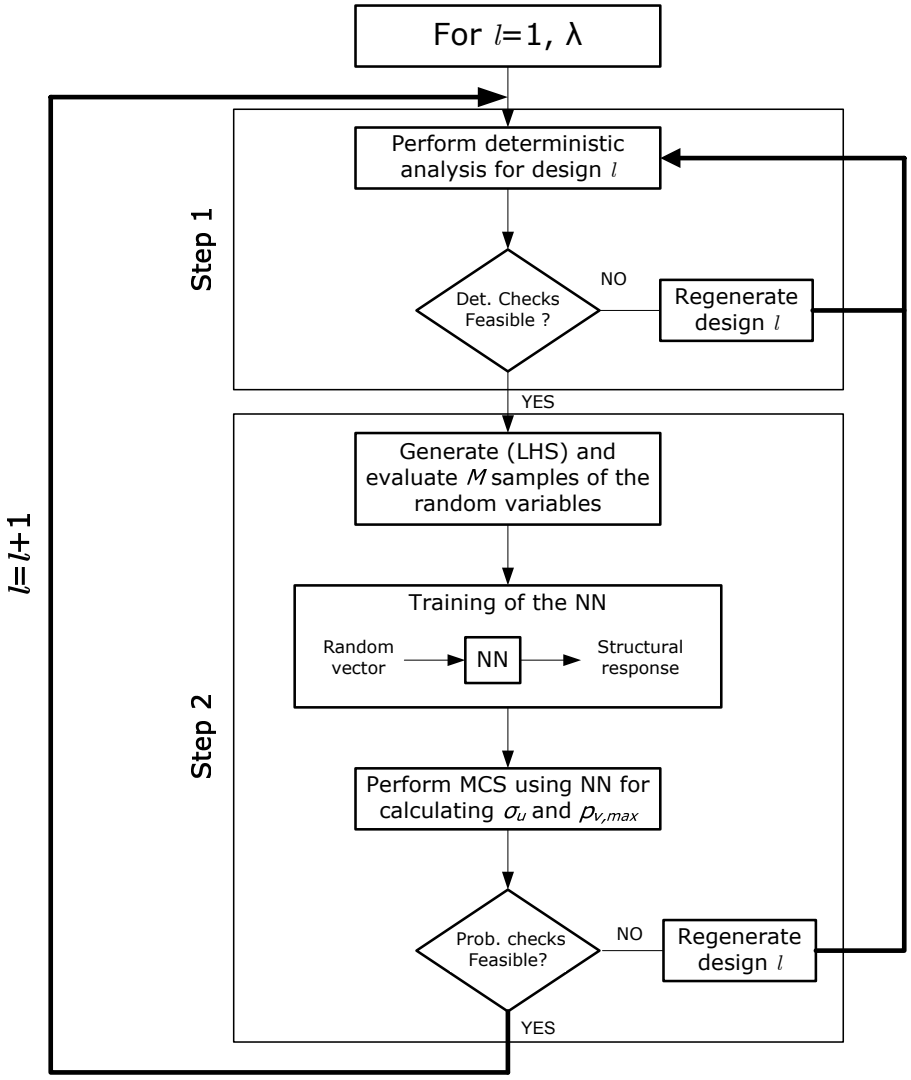


Figure 6.8 The NN assisted MCS procedure.

Figure 6.8 describes the procedure for assessing the λ offspring of an optimization cycle (generation) of CEA. After training, over a sample of N random vectors generated using LHS, the trained NN is applied as a predictor of the structural performance for every new vector of random variables encountered during the MCS. As soon as the NN is trained, the NN approximations of the structural response can be obtained.

Thus, the computational cost required for performing the probabilistic constraints' checks and calculating the statistical quantities needed by the RRDO formulation is reduced significantly as will be shown in detail in Chapter 8 (Numerical Applications – Probabilistic Optimization).

To avoid the problem of NN performing extrapolation instead of interpolation, described in Section 5.10.1, it is important to have sufficient and properly distributed, over the range of interest, training data. This possible drawback is alleviated by generating the training sample using the Latin Hypercube Sampling method in the range of $\mu_i \pm 6\sigma_i$ for

any random variable, where μ_i and σ_i are the mean value and standard deviation of the i-th random variable. This methodology ensures that 99.9999998% of the samples will be in this range (Koch et al. 2004) and thus NN will always predict interpolated values. If during the NN based Monte Carlo Simulation, a sample is generated outside the prescribed range then this sample is rejected and a new one is generated.

Chapter 7

7 Numerical Applications – Part A: Deterministic Optimization

This chapter contains the first part (Part A) of the numerical applications investigated in this thesis, where deterministic optimization is considered. In this chapter, five test examples are examined in total. The chapter is divided into two sections:

i. In the first section (Section 7.1), two multi-objective optimization test examples are considered, using either *Evolution Strategies* (ES) combined with standard multi-objective optimization methods or the proposed *Evolution Strategy for Multi-objective Optimization* (ESMO) algorithm for solving the multi-objective optimization problem.

ii. In the second section (Section 7.2), three single-objective examples are considered, using the proposed *Particle Swarm Optimization* (PSO) methodology and the proposed hybrid PSO-SQP methodology for solving the constrained structural optimization problem.

7.1 Multi-objective optimization

The first part of the deterministic optimization section includes two multi-objective optimization test examples, a multi-layered space truss under static loading and a six-story space frame under combined static and dynamic loading. The conclusions for both test examples are given in Section 7.1.3.

7.1.1 Multi-layered space truss

The optimum design with multiple objectives of a long span three layered aircraft hangar is investigated (Papadrakakis et al. 2002a). The objective functions considered for the problem are the weight of the structure and the maximum deflection, both to be minimized. This hangar is a triple layer space truss with a 5-layered front girder spanning over 130.9m. The front girder is formed by adding two layers, one at the top and the oth-

er at the bottom of the space truss, as shown in Figure 7.2. The heavily stressed top and bottom layers are designated as 'flanges' consisting of longitudinal and cross-girders, which are closed box sections. The diagonal members connecting the top and bottom flanges to the top and bottom chords of the triple-layer space frame are also closed box sections.

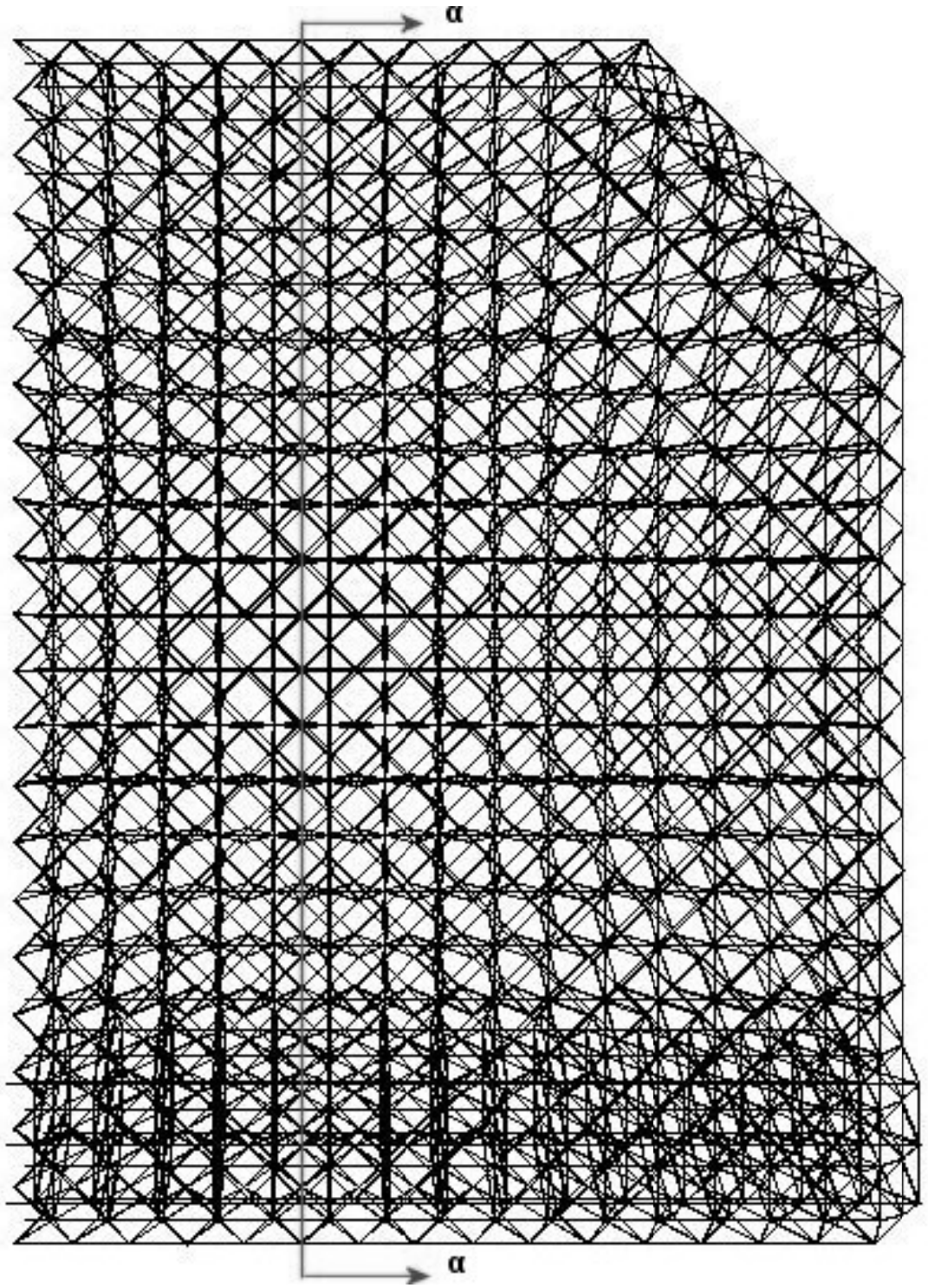


Figure 7.1 Multi-objective optimization - Multi-layered space truss: 3D model for the half of the real structure.

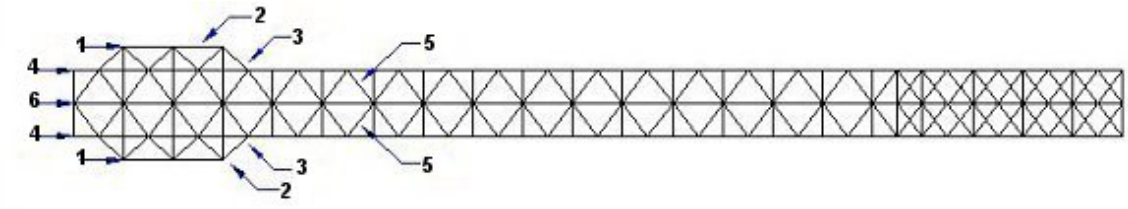


Figure 7.2 Multi-objective optimization - Multi-layered space truss: Cross-section of the space hangar.

The model has 3614 nodes, 10638 *Degrees Of Freedom* (DOFs) and 12974 members. Members of groups 1 to 3 are to be selected from the structural sections listed in Table 7.1 and members of groups 4 to 6 from the tube sizes given in Table 7.2. Taking advantage of the symmetry of the structure, the formulation of the problem was made for one half of the hangar, depicted in Figure 7.1, which results in a model with 5269 DOFs.

The members of the space truss are grouped (Figure 7.2) as follows:

1. Group 1: Longitudinal members of the top and bottom flanges.
2. Group 2: Cross girders of the top and bottom flanges.
3. Group 3: Bracing diagonals connecting the top and bottom flanges to the top and bottom chords of the space hangar.
4. Group 4: Top and bottom chords of the space truss.
5. Group 5: Diagonal bracing members connecting the top and bottom chords to middle chords.
6. Group 6: Middle chords of the space hangar.

Space truss structures usually have the topology of single or multi-layered flat or curved grids that can be easily constructed in practice. In the optimal design of trusses the constraints are the member stresses, nodal displacements, or frequencies. The stress constraints can be written as

$$\sigma_{\max} \leq \sigma_a \tag{7.1}$$

$$\sigma_a = \frac{\sigma_y}{\gamma_{\mathrm{M1}}} \tag{7.2}$$

where $\sigma_{\max}$ is the maximum axial stress in each element group (absolute value) for all loading cases, σ_a is the allowable axial stress and σ_y is the yield stress of the material. The safety factor γ_{M1} is a Eurocode 1 (CEN 1991) box value usually taken equal to 1.10. Similarly, the displacement constraints can be written as

$$|d| \leq d_{\mathrm{a}} \tag{7.3}$$

where d_{a} is the limit value of the displacement at a certain node or the maximum nodal displacement.

Euler buckling is also considered as a stress-type constraint in truss structures. This is enforced when the magnitude of a member's compressive stress is greater than a critical stress which usually is taken as the first buckling mode of a pin-connected member:

$$\sigma_{\min} \geq \sigma_b \tag{7.4}$$

$$\sigma_{\mathrm{b}} = \frac{P_{\mathrm{b}}}{A} = -\frac{1}{A}\left(\frac{\pi^2 EI}{L^2}\right) \tag{7.5}$$

where σ_{b} is the critical Euler buckling stress (with negative sign), $\sigma_{\min}$ is the minimum axial force (with negative sign or the maximum compressive axial force in absolute values), P_{b} is the compressive buckling axial force (with negative sign), I is the moment of inertia and L is the member length.

Table 7.1 Multi-objective optimization - Multi-layered space truss: Properties of the structural members (Database 1).

Section number	Type	Description
1	ISMC 100	Single Channel
2 - 12	2 x ISMC (75,100,125,150,175,200,225,250,300,350,400)	Closed box section made-up of 2 channels
13 - 16	2 x ISMC 400 with 2 x (8,12,16,25mm) thick MS Plates	Closed box section made-up of 2 channels With 2 plates welded at top and bottom
17	4 x ISMC 400	Closed double box section made-up of 4 channels
18 - 22	4 x ISMC 400 with 2 x (8,16,20,25,32mm) thick MS Plates	Closed double box section made-up of 4 channels with 2 plates welded at top and bottom
23 - 27	4 x ISMC 400 with 4 x (20,25,32,40,50mm) thick MS Plates	Closed double box section made-up of 4 channels with 4 plates welded at top and bottom

Table 7.2 Multi-objective optimization - Multi-layered space truss: Properties of the structural members (Database 2).

Section number	Outer diameter	thickness	Area (mm^2)
1	60.30	3.25	582.73
2	76.10	4.50	1012.63
3	88.90	4.85	1281.16
4	114.30	5.40	1848.19
5	139.70	5.40	2279.26
6	152.40	5.40	2494.8
7	165.10	5.40	2710.34
8	193.70	5.90	3482.35
9	219.10	5.90	3953.34
10	273.00	5.90	4952.8

The performance of the *Linear Weighting Method* (LWM) of Section 4.7.1 together with the results of the *Distance Function Method* (DFM) of Section 4.7.2 with p=1, 2 and 8 and the proposed ESMO method of Section 4.10.2 for the case of the simultaneous minimization of weight and maximum displacement are depicted in Figure 7.3.

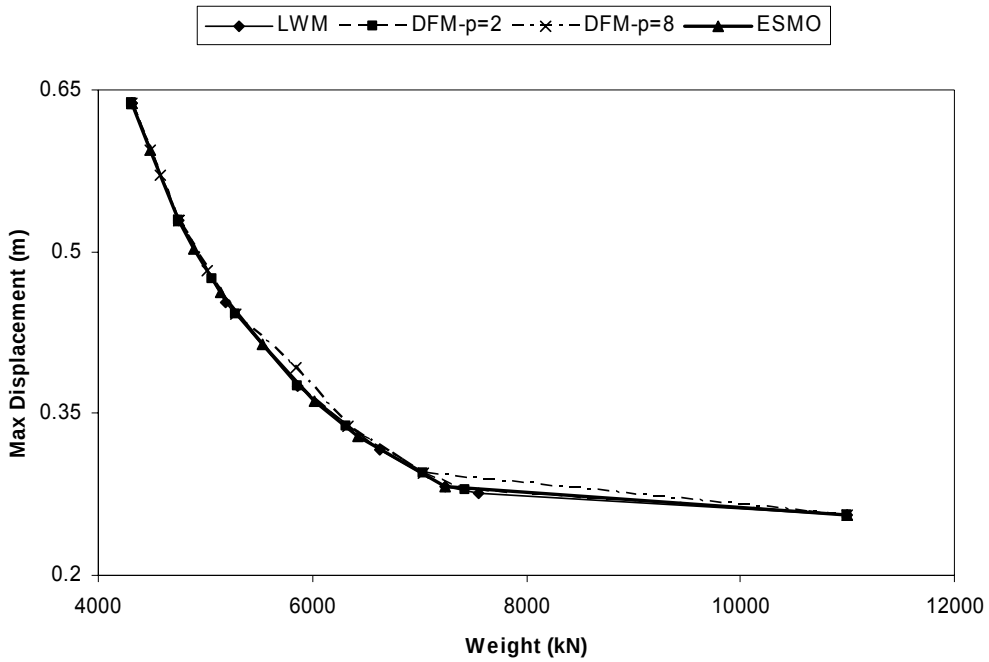


Figure 7.3 Multi-objective optimization - Multi-layered space truss: LWM, DFM and ESMO methods.

Figure 7.4 depicts the performance of the *Constraint Method* (CM), decribed in detail in Section 4.7.3, for the simultaneous minimization of the weight and the maximum displacement.

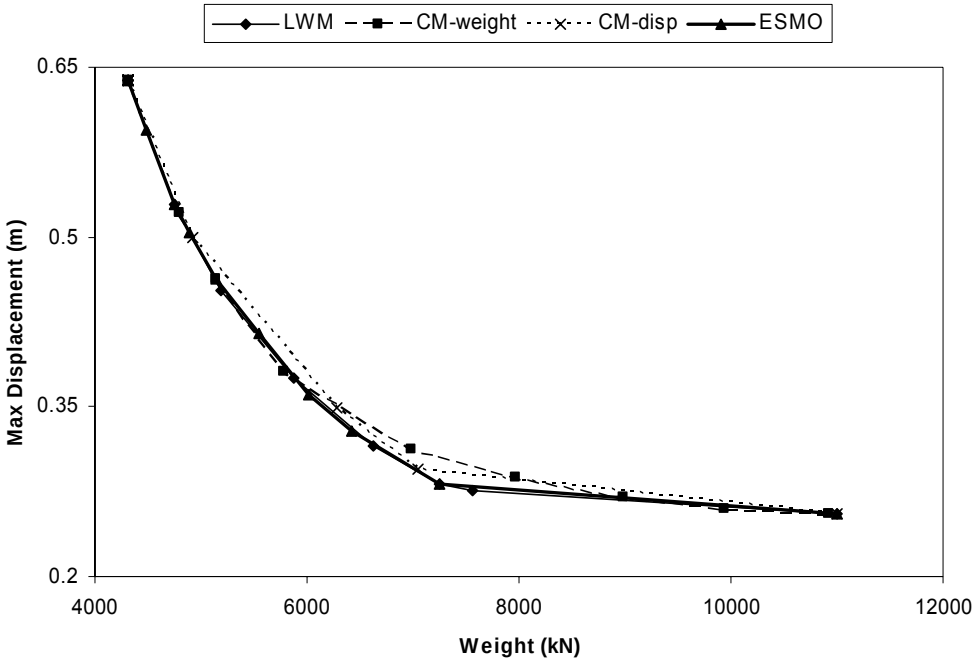


Figure 7.4 Multi-objective optimization - Multi-layered space truss: LWM, CM and ESMO methods.

Two cases are considered for the CM:

i. The weight as the only criterion and the maximum displacement as a constraint.

ii. The maximum displacement as the only criterion and the weight as a constraint.

These sets of Pareto optimal solutions are produced for different upper limits of the maximum displacement and for different upper limits of the weight of the structure. The proposed ESMO algorithm gives almost identical results compared to those obtained by the standard methods such as the linear weighting, distance function and constraint methods, as can be seen in Figures 7.3 and 7.4.

The performance of the LWM and the ESMO methods in terms of computational effort (no. of generations, no. of finite element analyses and CPU time) are presented in Table 7.3.

Table 7.3 Multi-objective optimization – Multi-layered space truss: Computational performance of the LWM and ESMO methods.

Method	Generations	FE analyses	CPU-Time (sec)
LWM	195	1163	7917
ESMO	28	312	2119

In terms of computational efficiency, it appears that all three standard methods considered require similar computational effort with approximately the same number of gen-

eration steps. On the other hand, as can be seen in Table 7.3, there is a substantial improvement in the computing time as the proposed ESMO method requires almost one order of magnitude less computing time than the standard methods.

7.1.2 Six story space frame under dynamic loading

The performance of the multi-objective optimization method proposed in the present thesis is investigated in a six story space frame benchmark test example (Papadrakakis et al. 2002a; Papadrakakis et al. 2002b). The equations of equilibrium for a finite element system in motion for the *i*-th design vector can be written in the usual form

$$\boldsymbol{M}(\boldsymbol{x}^i)\cdot\ddot{\boldsymbol{u}}(t)+\boldsymbol{C}(\boldsymbol{x}^i)\cdot\dot{\boldsymbol{u}}(t)+\boldsymbol{K}(\boldsymbol{x}^i)\cdot\boldsymbol{u}(t)=\boldsymbol{R}(t) \tag{7.6}$$

where $\boldsymbol{M}(\boldsymbol{x}^i)$, $\boldsymbol{C}(\boldsymbol{x}^i)$, and $\boldsymbol{K}(\boldsymbol{x}^i)$ are the mass, damping and stiffness matrices for the *i*-th design vector $\boldsymbol{x}^i$, $\boldsymbol{R}(t)$ is the external load vector, while $\boldsymbol{u}(t),\dot{\boldsymbol{u}}(t)$ and $\ddot{\boldsymbol{u}}(t)$ are the displacement, velocity, and acceleration vectors of the finite element assemblage for time *t*, respectively. Two design approaches will be considered:

i. Based on the *Direct Time Integration* (DTI) of the equations of motion;

ii. Based on the *Response Spectrum Modal Analysis* (RSMA), and the mode superposition approach (Papadrakakis et al. 2000a; Papadrakakis et al. 2000b; Papadrakakis et al. 2001a).

Direct Time Integration method

The Newmark integration scheme, an implicit method, is adopted in the present study to perform the DTI of the equations of motion. The equilibrium condition of Eq. (7.6) is discretized in time as follows

$$\boldsymbol{M}(\boldsymbol{x}^i)\cdot\ddot{\boldsymbol{u}}(t+\Delta t)+\boldsymbol{C}(\boldsymbol{x}^i)\cdot\dot{\boldsymbol{u}}(t+\Delta t)+\boldsymbol{K}(\boldsymbol{x}^i)\cdot\boldsymbol{u}(t+\Delta t)=\boldsymbol{R}(t+\Delta t) \tag{7.7}$$

where the variation of velocity and displacement are given by

$$\dot{\boldsymbol{u}}(t+\Delta t)=\dot{\boldsymbol{u}}(t)+\left[(1-\delta)\ddot{\boldsymbol{u}}(t)+\delta\cdot\ddot{\boldsymbol{u}}(t+\Delta t)\right]\Delta t \tag{7.8}$$

$$\boldsymbol{u}(t+\Delta t)=\boldsymbol{u}(t)+\dot{\boldsymbol{u}}(t)\cdot\Delta t+\left[\left(\frac{1}{2}-\alpha\right)\ddot{\boldsymbol{u}}(t)+\alpha\cdot\ddot{\boldsymbol{u}}(t+\Delta t)\right]\Delta t^2 \tag{7.9}$$

where α and δ are parameters that are determined in order to obtain integration accuracy and stability. Characteristic values of these parameters $a=1/2$ and $1/6\le\beta\le1/4$ that give good results in terms of accuracy and stability. Solving for $\ddot{\boldsymbol{u}}(t+\Delta t)$ in terms of $\boldsymbol{u}(t+\Delta t)$ from Eq. (7.9) and substituting for $\ddot{\boldsymbol{u}}(t+\Delta t)$ in Eq. (7.8) we obtain equations for $\ddot{\boldsymbol{u}}(t+\Delta t)$ and $\dot{\boldsymbol{u}}(t+\Delta t)$ in terms of the unknown displacements $\boldsymbol{u}(t+\Delta t)$ only. These two relations for $\ddot{\boldsymbol{u}}(t+\Delta t)$ and $\dot{\boldsymbol{u}}(t+\Delta t)$ are then substituted into Eq. (7.7) to solve for

$\boldsymbol{u}(t+\Delta t)$. As a result of this substitution the following well-known equilibrium equation is obtained at each time step

$$\boldsymbol{K}^{\text{eff}}(\boldsymbol{x}^i)\cdot\boldsymbol{u}(t+\Delta t) = \boldsymbol{R}^{\text{eff}}(t+\Delta t) \tag{7.10}$$

Creation of artificial accelerograms

The selection of the proper external loading vector $\boldsymbol{R}(t)$ to perform structural analyses under seismic loading conditions for design purposes is not an easy task due to the uncertainties involved in the seismic loading. For this reason a rigorous treatment of the seismic loading is to assume that the structure is subjected to a set of real and/or artificial earthquakes that are more likely to occur in the region where the structure is located. These artificial seismic excitations are produced as a series of artificial accelerograms compatible with the elastic design response spectrum of the region.

In this work the implementation published by Taylor (1989) for the generation of statistically independent artificial acceleration time histories is adopted. This method is based on the fact that any periodic function can be expanded into a series of sinusoidal waves

$$x(t) = \sum_k A_k \sin(\omega_k \cdot t + \phi_k) \tag{7.11}$$

where A_k is the amplitude, ω_k is the cyclic frequency and ϕ_k is the phase angle of the k-th contributing sinusoid. By fixing an array of amplitudes and then generating different arrays of phase angles, different motions can be generated which are similar in general appearance but different in the "details". The computer uses a random number generator subroutine to produce strings of phase angles with a uniform distribution in the range $[0, 2\pi]$. The amplitudes A_k are related to the spectral density function in the following way

$$G(\omega_k)\cdot\Delta\omega = \frac{{A_k}^2}{2} \tag{7.12}$$

where $G(\omega_k)\cdot\Delta\omega$ may be interpreted as the contribution to the total power of the motion from the sinusoid with frequency ω_k. The power of the motion produced by Eq. (7.11) does not vary with time. To simulate the transient character of real earthquakes, the steady-state motions are multiplied by a deterministic envelope function $I(t)$

$$Z(t) = I(t)\cdot\sum_k A_k \sin(\omega_k t + \phi_k) \tag{7.13}$$

The resulting motion is stationary in frequency content with peak acceleration close to the target peak acceleration. In this study a trapezoidal intensity envelope function is used. The generated peak acceleration is artificially modified to match the target peak acceleration, which corresponds to the chosen elastic design response spectrum. An it-

erative procedure is then implemented to smooth the calculated spectrum and improve the matching (Taylor 1989).

Response Spectrum Modal Analysis method

The RSMA method is based on a simplification of the mode superposition approach with the aim to avoid time history analyses which are required by both the direct integration and mode superposition approaches. In the case of the response spectrum modal analysis, Eq. (7.7) is modified according to the modal superposition approach, for the *i*-th design vector, in the following form

$$\bar{\boldsymbol{M}}(\boldsymbol{x}^i)\cdot\ddot{\boldsymbol{u}}(t)+\bar{\boldsymbol{C}}(\boldsymbol{x}^i)\cdot\dot{\boldsymbol{u}}(t)+\bar{\boldsymbol{K}}(\boldsymbol{x}^i)\cdot\boldsymbol{u}(t)=\bar{\boldsymbol{R}}(t) \tag{7.14}$$

where

$$\bar{\boldsymbol{M}}(\boldsymbol{x}^i)=\boldsymbol{\Phi}^{i\mathrm{T}}\cdot\boldsymbol{M}^i\cdot\boldsymbol{\Phi}^i \tag{7.15}$$

$$\bar{\boldsymbol{C}}(\boldsymbol{x}^i)=\boldsymbol{\Phi}^{i\mathrm{T}}\cdot\boldsymbol{C}^i\cdot\boldsymbol{\Phi}^i \tag{7.16}$$

$$\bar{\boldsymbol{K}}(\boldsymbol{x}^i)=\boldsymbol{\Phi}^{i\mathrm{T}}\cdot\boldsymbol{K}^i\cdot\boldsymbol{\Phi}^i \tag{7.17}$$

$$\bar{\boldsymbol{R}}(t)=\boldsymbol{\Phi}^{i\mathrm{T}}\cdot\boldsymbol{R}(t) \tag{7.18}$$

are the generalized values of the corresponding matrices and the loading vector, while $\boldsymbol{\Phi}^i$ is an eigenmode shape matrix to be defined in the following paragraphs. For simplicity the matrices $\boldsymbol{M}(\boldsymbol{x}^i)$, $\boldsymbol{C}(\boldsymbol{x}^i)$ and $\boldsymbol{K}(\boldsymbol{x}^i)$ are denoted by $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{K}$, respectively. These matrices correspond to the design, which is defined by the *i*-th vector of the design parameters. According to the modal superposition approach, the system of *N* simultaneous differential equations, is transformed into a set of *N* independent normal-coordinate equations.

In the RSMA method, a number of different formulas have been proposed to obtain reasonable estimates of the maximum response based on the spectral values without performing time history analyses for a considerable number of transformed dynamic equations. The simplest and most popular of these is the *Square Root of the Sum of Squares* (SRSS) of the modal responses. According to this estimate the maximum total displacement is approximated by

$$u_{\max}=\sqrt{u_{1,\max}^2+u_{2,\max}^2+\ldots+u_{N,\max}^2} \tag{7.19}$$

where $u_{j,\max}$ corresponds to the maximum displacement calculated from the *j*-th transformed dynamic equations over the complete time period. The use of Eq. (7.19) permits this type of "dynamic" design by knowing only the maximum modal displacements $u_{j,\max}$.

The following steps summarize the response spectrum modal analysis used in this work and in a number of seismic codes around the world. For every design vector *i*:

1. Calculate a number $m' < N$ of eigenfrequencies and the corresponding eigenmode shape matrices, which are classified in the following order $(\omega_1, \omega_2, \ldots, \omega_{m'})$, $\boldsymbol{\Phi} = [\boldsymbol{\varphi}^1, \boldsymbol{\varphi}^2, \ldots, \boldsymbol{\varphi}^{m'}]$, respectively, where ω_j and $\boldsymbol{\varphi}^j$ (a column vector) are the j-th eigenfrequency and eigenvector, respectively, m' is a user specified number, based on experience or on previous test analyses, which has to satisfy the requirement of *step 6*.
2. Calculate the generalized masses $\overline{m}_j$:

$$\overline{m}_j = \boldsymbol{\varphi}^{j\mathrm{T}} \cdot \boldsymbol{M} \cdot \boldsymbol{\varphi}^j \tag{7.20}$$

3. Calculate the coefficients L_j:

$$L_j = \boldsymbol{\varphi}^{j\mathrm{T}} \cdot \boldsymbol{M} \cdot \boldsymbol{r} \tag{7.21}$$

where $\boldsymbol{r}$ is the influence vector, which represents the displacements of the masses resulting from static application of a unit ground displacement.

4. Calculate the modal participation factor Γ_j:

$$\Gamma_j = \frac{L_j}{\overline{m}_j} \tag{7.22}$$

5. Calculate the effective modal mass for each design vector and for each eigenmode:

$$m_{\mathrm{eff},j} = \frac{(L_j)^2}{\overline{m}_j} \tag{7.23}$$

6. Calculate a number $m < m'$ of the important eigenmodes. According to Eurocode (CEN 1998) the minimum number of eigenmodes that has to be taken into account is defined by the assumption that the sum of the effective eigenmasses must not be less than the 90% of the total vibrating mass m_{tot} of the system. Thus the first m eigenmodes that satisfy the equation

$$\sum_{j=1}^{m} m_{\mathrm{eff},j} \geq 0.90 \cdot m_{tot} \tag{7.24}$$

are taken into consideration.

7. Calculate the values of the spectral acceleration $R_d(T_j)$ that correspond to each eigenperiod T_j of the m important modes from the response spectrum of the region. The eigenperiods are calculated by

$$T_j = \frac{2\pi}{\omega_j} \tag{7.25}$$

8. Calculate the modal displacements according to equation

$$(SD)_j = \frac{R_d(T_j)}{\omega_j{}^2} = \frac{R_d(T_j)\cdot T_j{}^2}{4\pi^2}, \quad j=1,\ldots,m \tag{7.26}$$

9. Calculate the maximum modal displacements from

$$u_{j,\max} = \Gamma_j \cdot \varphi^j \cdot (SD)_j, \quad j=1,\ldots,m \tag{7.27}$$

10. The total maximum displacement is calculated by superimposing the maximum modal displacements according to Eq. (7.19).

The elastic design response spectrum considered in the study is depicted in Figure 7.5 for damping ratio ξ=2.5%, together with the corresponding response spectrum of the first out of five artificial accelerograms. The five uncorrelated artificial accelerograms shown in Figure 7.6, produced by the procedure described previously based on the elastic design response spectrum, are used as the input seismic excitation for the Direct Time Integration approach of the numerical tests.

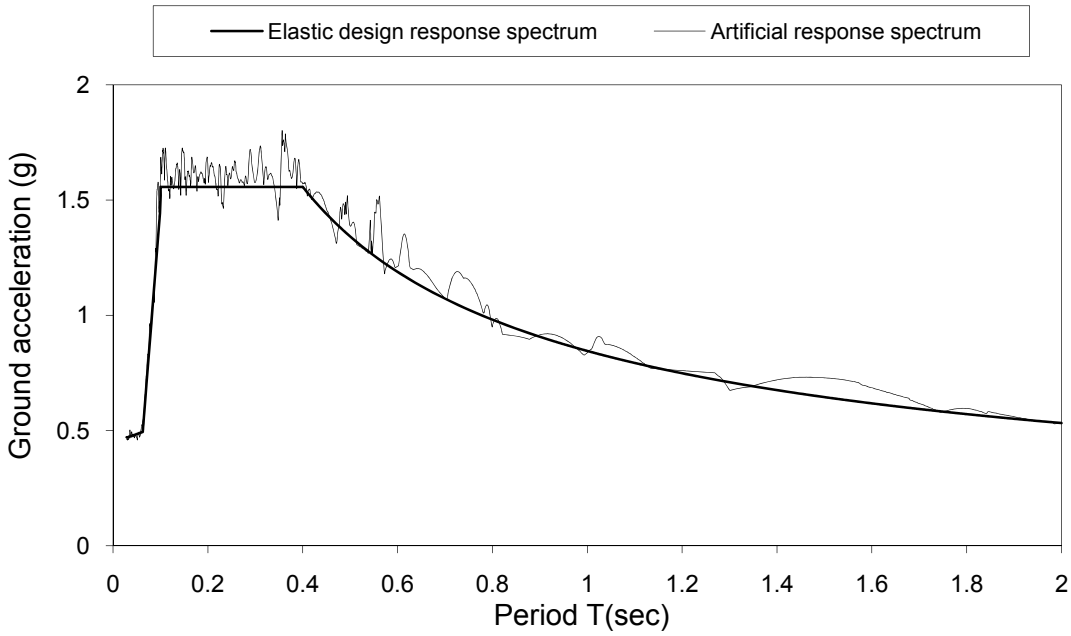


Figure 7.5 Multi-objective optimization - Six story space frame: Elastic design response spectrum of the region and response spectrum of the first artificial accelerogram (ξ=2.5%).

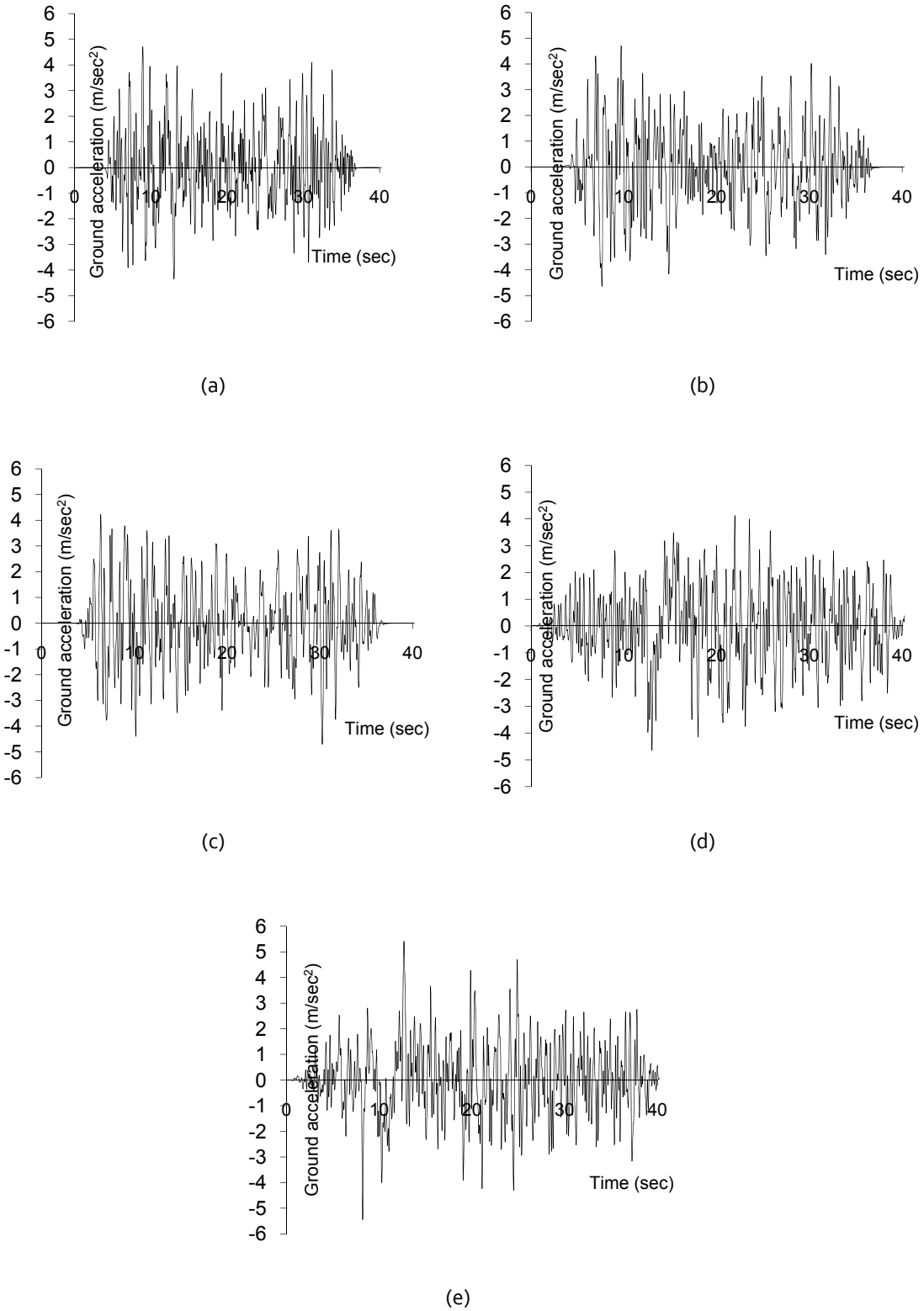


Figure 7.6 Multi-objective optimization - Six story space frame:
The five artificial accelerograms.

In the optimal design of frames the constraints are usually the member stresses and nodal displacements or inter-story drifts. For rigid frames with I-shapes, the stress constraints under allowable stress design requirements specified by Eurocode 3 (CEN 1993), are expressed by the following formula

$$q = \frac{N_{sd}}{Af_y / \gamma_{M1}} + \frac{M_{sd,y}}{W_{pl,y} f_y / \gamma_{M1}} + \frac{M_{sd,z}}{W_{pl,z} f_y / \gamma_{M1}} \leq 1.0 \tag{7.28}$$

where N_{sd}, $M_{sd,y}$, $M_{sd,z}$ are the stress resultants, $W_{pl,y}$, $W_{pl,z}$ are the plastic first moments of inertia, f_y is the yield stress and A is the cross section area. The safety factor γ_{M1} is a Eurocode 1 (CEN 1991) box value usually taken equal to 1.10.

The objective functions considered for this problem are the weight of the structure, the maximum displacement and the first eigenperiod. The weight and the maximum displacement are to be minimized, while the first eigenperiod is to be maximized. The constraints are imposed on the inter-story drifts and for each element group on the maximum non-dimensional ratio q of Eq. (7.28) under a combination of axial force and bending moments.

The space frame consists of 63 elements with 180 DOFs as shown in Figure 7.7. The length of the beams and the columns are L_1=7.32 m and the columns L_2=3.66 m, respectively. The structure is loaded with a 19.16 kPa gravity load on all floor levels and a static lateral load of 109 kN applied at each node in the front elevation along the z direction.

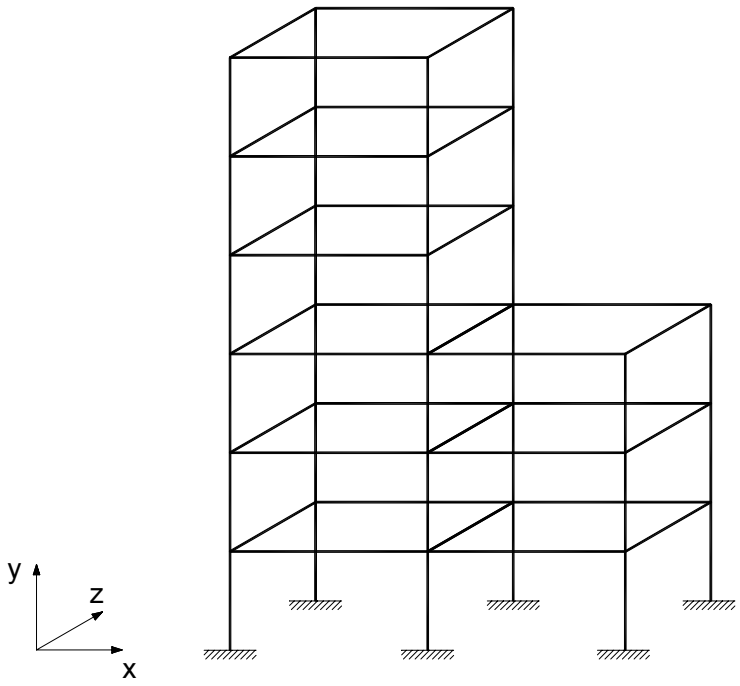


Figure 7.7 Multi-objective optimization - Six story space frame: 3D model of the structure.

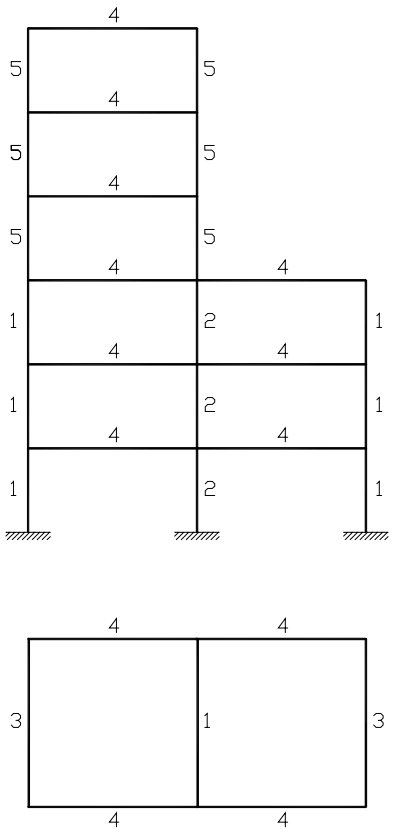


Figure 7.8 Multi-objective optimization - Six story space frame: Element groups.

The element members are divided into 5 groups, as shown in Figure 7.8, each one having two design variables resulting in ten design variables in total. The cross section of each member is assumed to be an I-shape and for each member two design variables are assigned as shown in Figure 7.9. The modulus of elasticity is $E=200\text{GPa}$ and the yield stress is $\sigma_y=250\text{MPa}$.

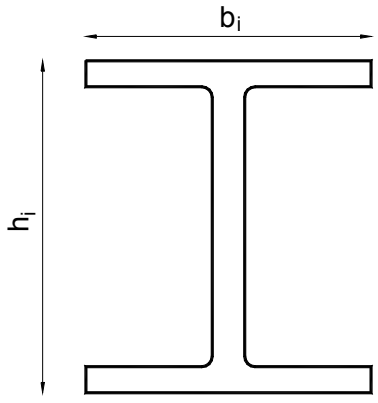


Figure 7.9 Multi-objective optimization - Six story space frame: I-shape cross section.

Weight and maximum displacement minimization

The Pareto optimal set of solutions for the case of seeking the simultaneous minimization of the weight and the maximum displacement was computed with LWM, CM and ESMO methods, for both static and combined static and seismic loading conditions.

In Figure 7.10 the performances of LWM, ESMO and the DFM methods are presented for the static case and the combined static and dynamic case with Direct Time Integration. For the case of the DFM the zero (**0**) point was considered as the utopian point, while four different schemes of the DFM were examined, p=1 (equivalent to the LWM), p=2 (called quadratic LWM) and p=8 (equivalent to the p=∞).

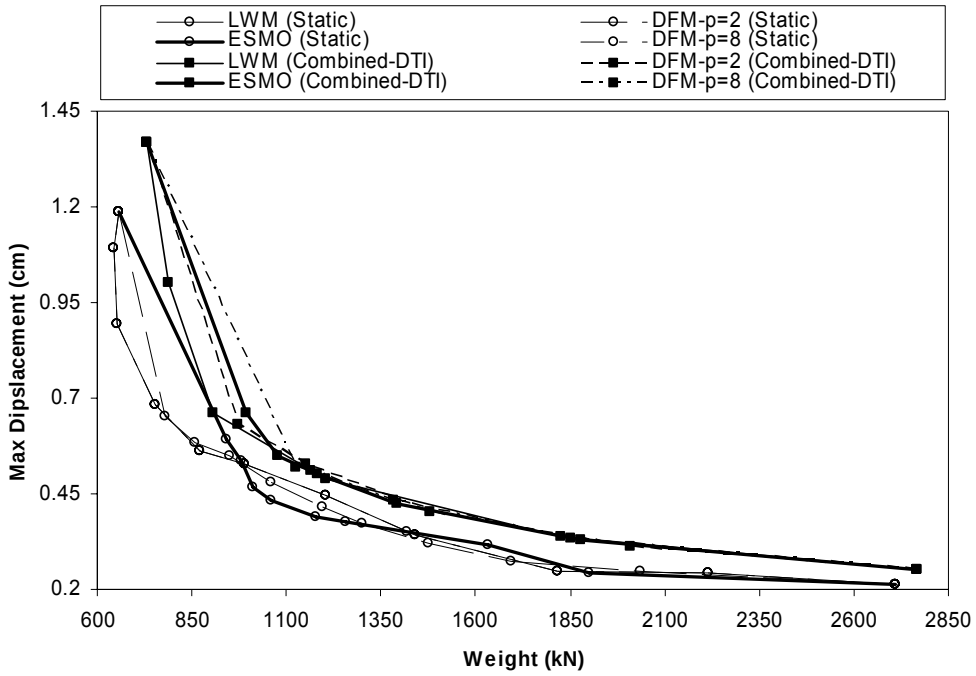


Figure 7.10 Multi-objective optimization - Six story space frame: Performance of the methods for static and combined static and seismic loading conditions.

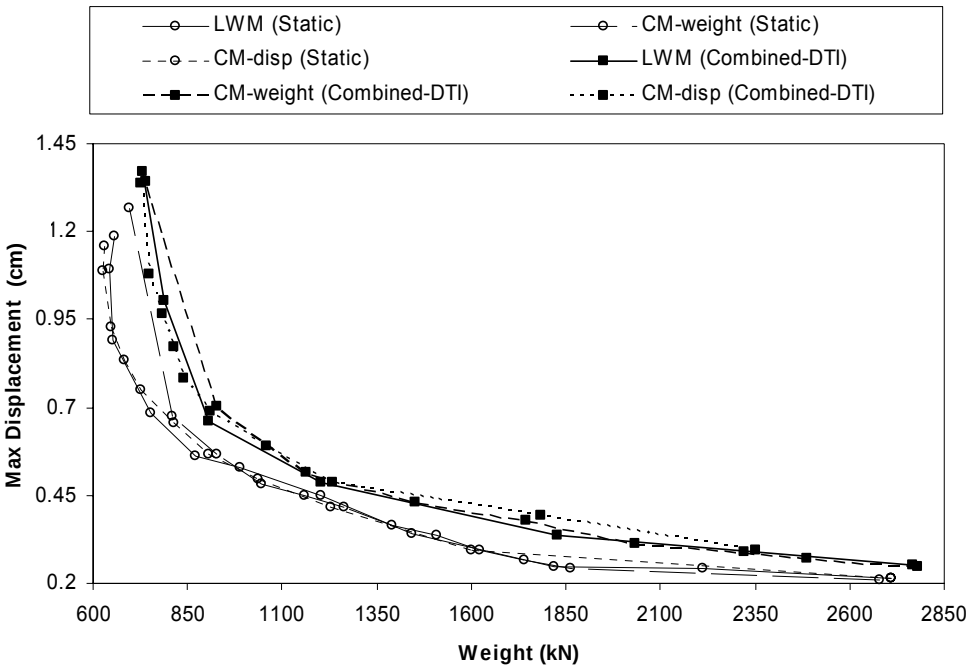


Figure 7.11 Multi-objective optimization - Six story space frame: Performance of the methods for static and combined static and seismic loading conditions.

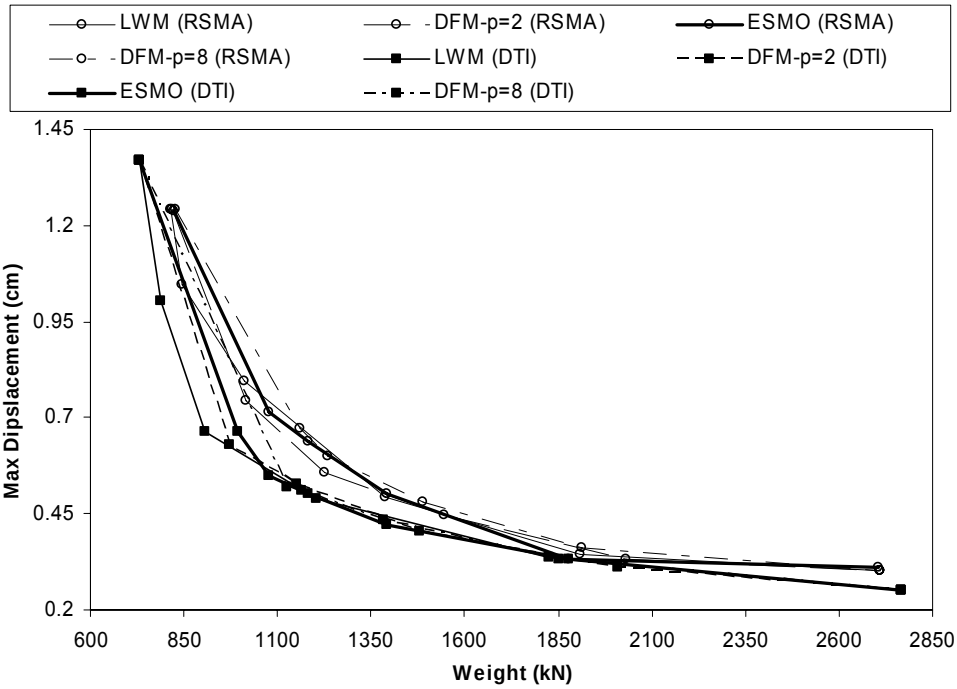


Figure 7.12 Multi-objective optimization - Six story space frame: Performance of the methods for combined static and seismic loading conditions.

In Figure 7.11 the performances of LWM and CM are presented for the static case and the combined static and dynamic case of DTI. The CM is implemented in the following two variations:

i. The weight as the only criterion and the maximum displacement as a constraint.

ii. The maximum displacement as the only criterion and the weight as a constraint.

In Figure 7.12 the performances of LWM, ESMO and the DFM methods are presented for both the combined static and dynamic cases with RSMA and DTI, for comparison purposes.

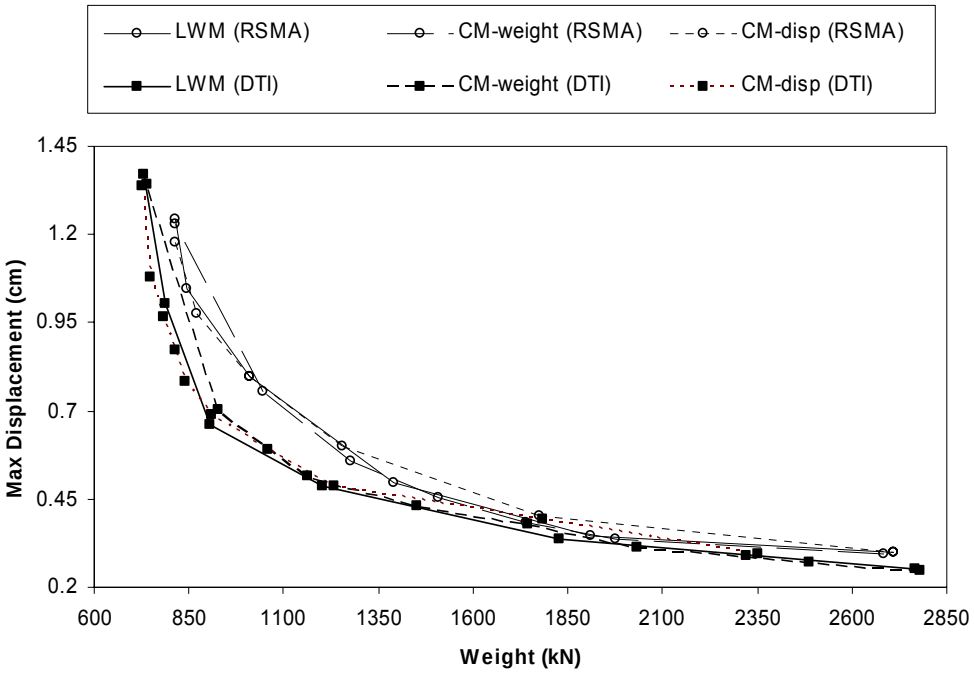


Figure 7.13 Multi-objective optimization - Six story space frame: Performance of the methods for combined static and seismic loading conditions.

In Figure 7.13 the performances of LWM and CM are presented for both the combined static and dynamic cases of RSMA and DTI. The CM is again implemented in two variations:

i. The weight as the only criterion and the maximum displacement as a constraint;

ii. The maximum displacement as the only criterion and the weight as a constraint.

Table 7.4 shows the performance of the methods in terms of computational efficiency for the combined static and dynamic loading case (no. of generations, no. of finite element analysis and CPU time).

Table 7.4 Multi-objective optimization - Six story space frame: Performance of the standard and ESMO methods for dealing with multi-objectives for dynamic loading conditions.

Method	Generations	FE analyses	CPU-Time (sec)
LWM (Combined-DTI)	372	2609	254112
ESMO (Combined-DTI)	28	367	35788
LWM (Combined-RSMA)	411	2901	109803
ESMO (Combined-RSMA)	31	401	15171

In Figures 7.12 and 7.13 it can be seen that the Pareto optimal solutions achieved by the DTI approach under the multiple loading conditions of the five artificial accelerograms given in Figure 7.6 are lower than the corresponding designs given by the RSMA. It can be concluded that the dynamic approach based on time history analyses gives more economic designs than the approximate RSMA, at the expense of requiring more computational effort.

Weight minimization and first eigenperiod maximization

The Pareto optimal set of solutions for the case of seeking the simultaneous minimization of the weight and the maximization of the first eigenperiod was computed with LWM, CM and ESMO methods, for static loading conditions.

In Figure 7.14 the performances of LWM, ESMO and the DFM methods are presented for the static case. For the case of the DFM the zero (**o**) point was considered as the utopian point, while three different schemes of the DFM were examined: (i) p=1, equivalent to the LWM; (ii) p=2, called quadratic LWM; and (iii) p=8, equivalent to the $p=\infty$.

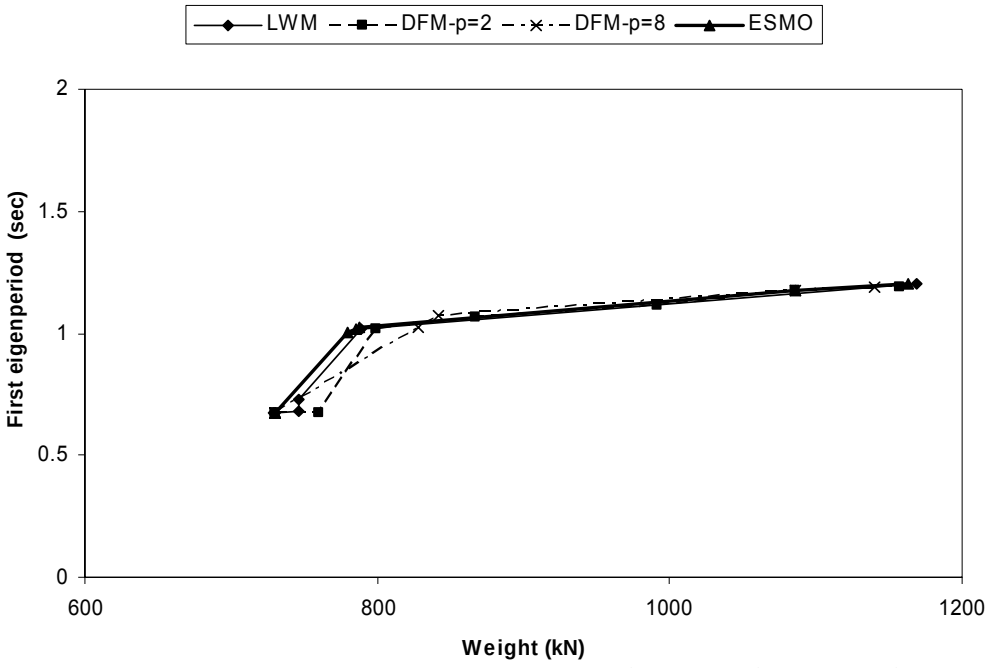


Figure 7.14 Multi-objective optimization - Six story space frame: Performance of Linear (p=1), Distance and ESMO methods.

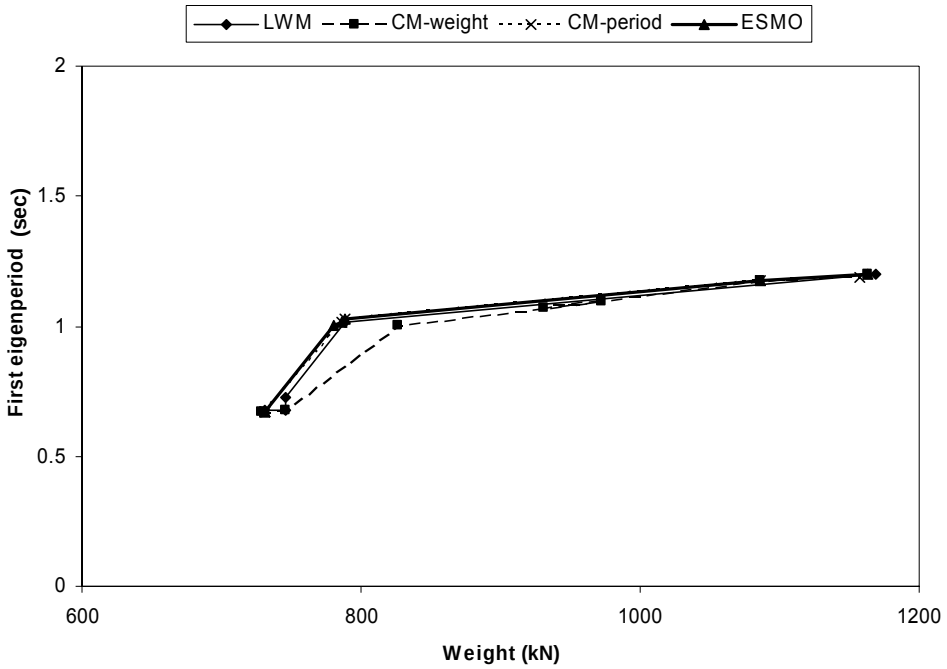


Figure 7.15 Multi-objective optimization - Six story space frame: Performance of Linear (p=1), Constraint and ESMO methods.

In Figure 7.15 the performances of LWM, CM and ESMO are presented for the static case. The CM is implemented in two variations:

i. The weight as the only criterion and the first eigenperiod as a constraint;

ii. The first eigenperiod as the only criterion and the weight as a constraint.

The proposed ESMO algorithm gives almost identical results compared to those obtained by the standard methods such as the linear weighting, distance function and constraint methods, as can be seen in Figures 7.14 and 7.15.

7.1.3 Conclusions on the multi-objective optimization test examples

The test examples show that Evolution Strategies and in particular the proposed ESMO algorithm can be considered as an efficient tool for multi-objective design optimization of space frames under static or combined static and dynamic loading. The proposed ESMO algorithm proved to be a robust and reliable optimization tool. The various standard methods, such as LWM, DFM, CM and the proposed ESMO have shown to produce almost equivalent results, in terms of the final Pareto front achieved.

In terms of computational efficiency, it appears that all three standard methods considered require similar computational effort with approximately the same number of generation steps. On the other hand there is a substantial improvement in the computing time for the ESMO case as the proposed methodology requires almost one order of magnitude less computing time than the standard methods.

The presented results also show that it is possible to achieve an optimal design under seismic loading and multiple objectives. Both design methodologies, DTI based on a number of artificially generated earthquakes and the RSMA adopted by the seismic codes have been implemented and compared. It can be concluded that the dynamic approach based on time history analyses gives more economic designs than the traditional approximate RSMA, at the expense of requiring more computational effort.

7.2 Particle Swarm Optimization (PSO)

The performance of the proposed PSO algorithm is examined in three benchmark test examples (Plevris et al. 2008b; Plevris and Papadrakakis 2009a; Plevris and Papadrakakis 2009b). The first one is a ten bar truss example with 10 design variables, the second is a 25 member space truss with 8 design variables, while the third is a 72 member space truss with 16 design variables.

The PSO scheme used in the study is the fully informed PSO described in Section 3.11, equipped with the non-linear inertia update rule described in Section 3.11.5 and the constraint handling technique of Section 3.11.4, unless otherwise stated. The conclusions for both test examples are given together in Section 7.2.3.

7.2.1 10 bar plane truss

This is the standard benchmark 10 bar plane truss shown in Figure 7.16 with the following structural characteristics: modulus of elasticity E=10000ksi, material weight ρ=0.1lb/in^3, length L=360in, load P=100kip. The structural members are divided into 10 groups. The design variables are the cross section areas of each member group in the interval [0.1, 35] (in^2). The constraints are imposed on stresses and displacements. The maximum allowable displacement in the $\pm x$ and $\pm y$ directions for each node is d_{max}=2in, while the maximum allowable stress (absolute value) is σ_{allow}=25ksi in tension or compression and the objective is to minimize the weight of the structure under the specified constraints.

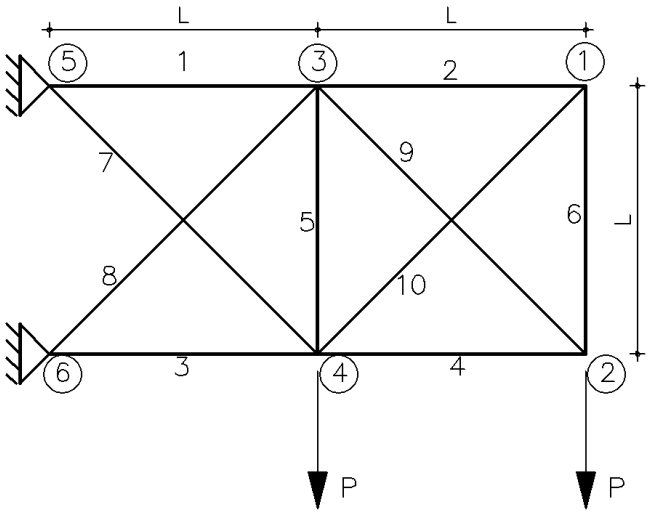


Figure 7.16 PSO - 10 bar plane truss: The truss model.

The influence of the inertia update rule on the optimization process of the PSO schemes will be first investigated. Three PSO schemes are considered, with a fixed inertia value $w=w_{\max}$ during all iterations, the linear update rule of Eq. (3.50) and the proposed non-linear update rule shown in Figure 3.12. The basic PSO used is shown in Table 7.5.

Table 7.5 PSO - 10 bar plane truss: PSO parameters used for the inertia update rule check.

Symbol	Value
NP	20
n	10
w	$w_{\max}=0.95$ $w_{\min}=0.5$
$\boldsymbol{x}^L, \boldsymbol{x}^U$	$x^L{}_i=0.1$ $x^U{}_i=35$ for all dimensions (in^2)
$\boldsymbol{v}^{\max}$	$v^{\max}{}_i=17.5$ for all dimensions (in^2)
c_1, c_2	$c_1=c_2=2$
$t_{\max}$	200

Ten PSO optimization runs are performed for each of the three cases. The results obtained for 200 PSO iterations are reported in Table 7.6 which shows the objective function values obtained for the best run, the worst run and the average of the 10 runs for each case.

Table 7.6 PSO - 10 bar plane truss:
Statistical results of the objective function for 10 PSO runs after 200 iterations.

Result	Fixed rule ($w = w_{max}$)	Linear rule	Non-linear cubic rule ($\alpha_w = 1.3$)
Best (lb)	5212.69	5111.72	5062.30
Worst (lb)	5573.70	5225.75	5121.46
Average (lb)	5396.36	5162.84	5098.77

It can be observed that the linear rule performs much better than the fixed rule, while the non-linear cubic rule improves further the result of the linear rule. Figure 7.17 depicts the convergence history for the three cases in terms of the average results over 10 optimization runs.

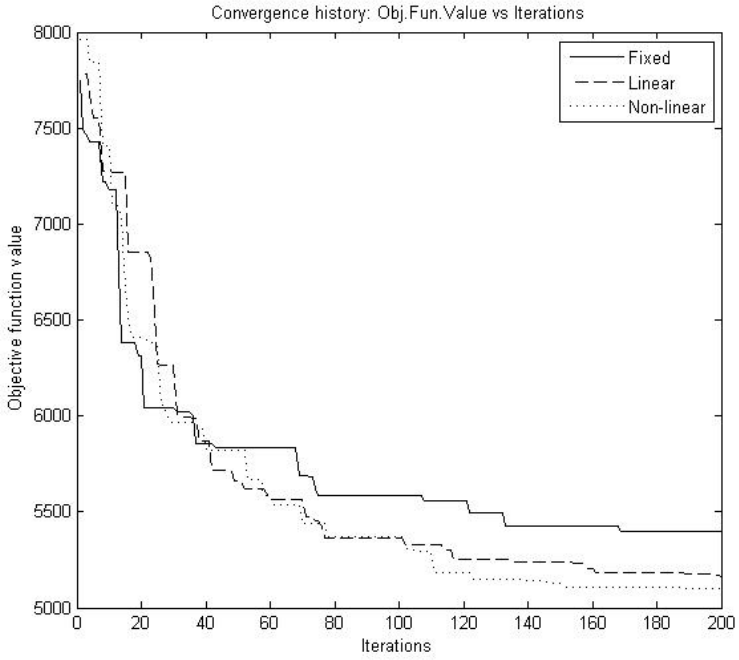


Figure 7.17 PSO - 10 bar plane truss: Convergence history for the three PSO schemes.

For this test example, the best design and the values of the constraints obtained by the non-linear rule are given in Tables 7.7 and 7.8, respectively.

Table 7.7 PSO - 10 bar plane truss: Optimum design obtained.

Property	Value
A1 (in^2)	30.9810
A2 (in^2)	0.1000
A3 (in^2)	23.1714
A4 (in^2)	15.6935
A5 (in^2)	0.1000
A6 (in^2)	0.5848
A7 (in^2)	7.4298
A8 (in^2)	20.6310
A9 (in^2)	21.3287
A10 (in^2)	0.1000
Weight (lb)	5062.30

Table 7.8 PSO - 10 bar plane truss: Feasibility of the optimum design.

Constraints	Value	Allowable value	Active
Max Stress (ksi)	24.9745	25	•
Max Displacement (in)	1.999998	2	•

It can be seen that for the above optimum design, all constraints have been met and are active, as the proposed constraint handling technique for the PSO always guarantees feasibility of the optimum design achieved.

The performance of the optimization algorithms is also studied with a convergence criterion connected to the improvement of the value of the objective function for a given number of iterations. If the relative improvement of the objective function over the last k_f=30 iterations is less or equal to f_m=10^{-6}, convergence is supposed to have been achieved.

The results reported in Table 7.9 include the average number of iterations needed for convergence and the objective function values obtained for the best run, the worst run and the average of 10 runs.

Table 7.9 PSO - 10 bar plane truss: Statistical results for 10 PSO runs.

Result	Fixed rule ($w = w_{max}$)	Linear rule	Non-linear rule ($\alpha_w = 1.3$)
Average number of iterations	122	172	185
Best (lb)	5356.71	5102.53	5065.24
Worst (lb)	5720.01	5443.08	5227.05
Average (lb)	5548.38	5259.88	5106.04

The non-linear rule performs much better than the other two PSO schemes in terms of the best result, the worst result and the average result. This is due to the fact that using the non-linear rule, convergence towards the optimum is smoother, and as a result the optimizer is more likely to find a better optimum solution before the termination criterion is satisfied. This is very important in practical structural optimization where the number of iterations needed for convergence is not known a priori.

Constraint handling technique investigation

In this study, the proposed linear segment penalty function constraint handling technique is compared with the death penalty approach (Hu et al. 2003) and the redirection approach (Venter and Sobieszczanski-Sobieski 2004), described in detail in Section 0. With the proposed technique, a penalty function proportional to the degree of the maximum violation of the constraints is applied to the objective function. In the death penalty approach, infeasible designs are ignored for the calculation of *PBest* or *Gbest*, which is equivalent to applying a very severe penalty to every infeasible design. In the redirection approach, infeasible designs are redirected closer to the feasibility boundary by resetting to zero the velocity vector $\boldsymbol{v}^j(t)$ for a particle j with violated constraints at iteration t.

The PSO parameters of the previous study were used, with the non-linear weight update rule. The redirection constraint handling technique failed to produce good quality of results, since it converged to infeasible solutions far from the feasibility boundary for a number of runs, while for other runs it converged to feasible solutions far from the optimum. For this reason the two other constraint handling techniques are compared which produce always feasible optimum designs and good quality of results. The convergence history of these two techniques is depicted in Figure 7.18.

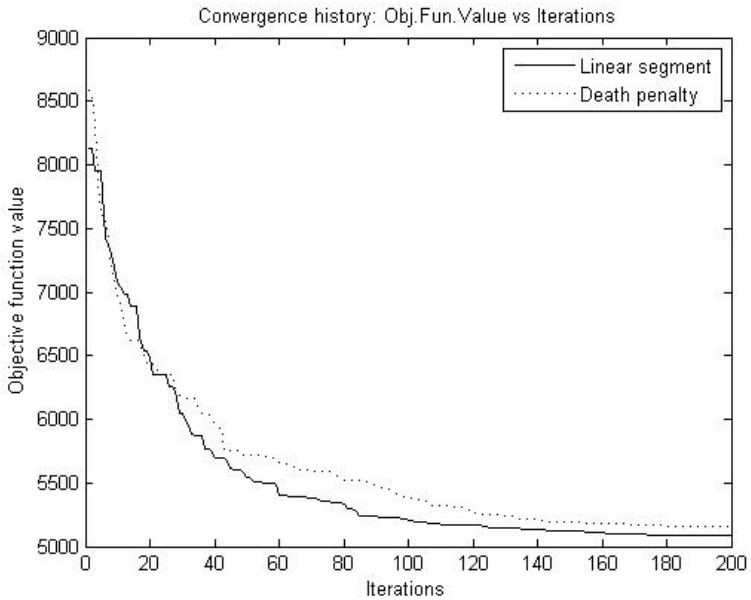


Figure 7.18 PSO - 10 bar plane truss:
Convergence history for the two PSO constraint handling techniques.

It can be seen that both methods converge to the optimum, while the convergence rate of the proposed constraint handling method is better. This is due to the fact that the proposed method takes into account the infeasible designs as well, which is beneficial for the convergence behavior of the optimization algorithm (Michalewicz 1995).

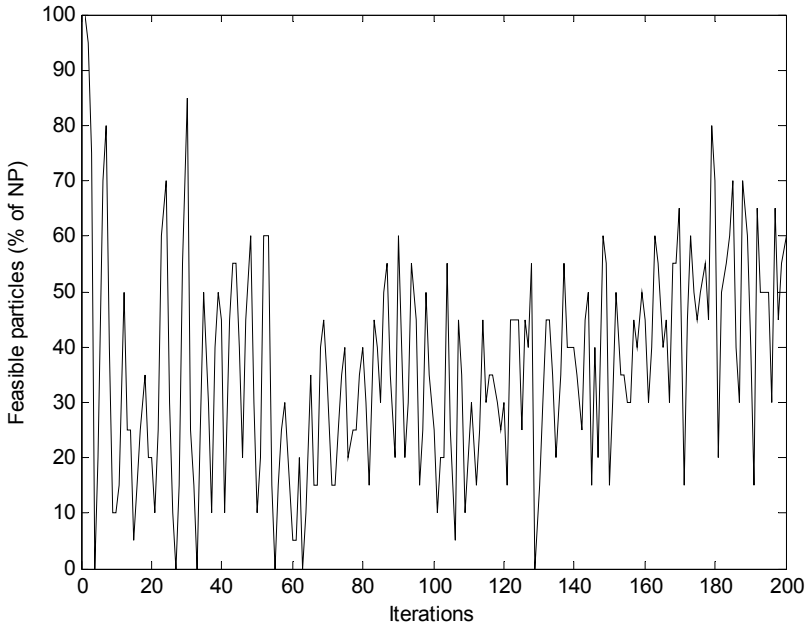


Figure 7.19 PSO - 10 bar plane truss:
Ratio of feasible particles in the population.

Figure 7.19 depicts the ratio of feasible particles in the population, throughout the PSO iterations, for the linear segment penalty function approach. It can be seen that the ratio varies widely with the iterations and that the algorithm always tries to improve the ratio once it reaches lower values of 10% - 30%. At the end of the optimization process, the value of the ratio improves, reaching 60%.

The hybrid PSO-SQP method

For the hybrid PSO-SQP scheme a relaxed termination criterion is applied for the PSO before SQP takes over the search for the optimum. The PSO is mainly used to explore the design space, detect the neighborhood of the global optimum and provide a good starting design point for the SQP phase.

First we apply the SQP optimizer, for various initial designs. Four initial designs have been selected, corresponding to design variables values 35, 25, 15 and 5 for every dimension of the problem, resulting to the objective function values given in Table 7.10.

Table 7.10 PSO - 10 bar plane truss: Convergence behavior of SQP.

Starting point (in^2)	Iterations	Obj. fun. evaluations	Obj. function value (lb)
"35"	17	210	5473.62
"25"	14	181	5473.62
"15"	15	184	5179.48
"5"	21	250	5179.48

It can be seen that the SQP method converges to suboptimal solutions, as a good model initialization is always required for SQP to produce good results.

Next, we implement the PSO-SQP scheme. If the relative improvement of the objective function over the last k_f=15 iterations of the PSO optimizer is less or equal to a threshold value (in this case f_m=10^{-4}), the PSO phase is terminated and subsequently the SQP starts from the best estimate of the PSO. The SQP termination criterion is connected to the first order optimality measure for constrained optimization, in terms of the infinite norm. If the magnitude of directional derivative in the search direction is less than a tolerance value of 10^{-5} and there is no constraint violation, convergence has been achieved.

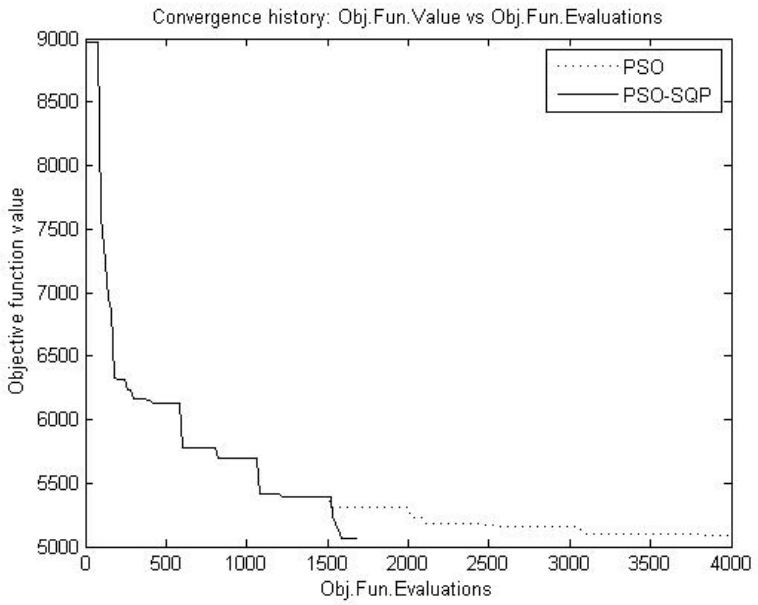


Figure 7.20 PSO - 10 bar plane truss: Convergence history for the hybrid PSO-SQP scheme.

The convergence history of the hybrid PSO-SQP scheme, compared to the basic PSO scheme is depicted in Figure 7.20 in terms of objective function value and objective function evaluations. In the hybrid scheme, the PSO needs 1520 function evaluations to reach an objective function value of 5395.43 and SQP needs another 169 objective function evaluations to converge to an optimum value of 5060.85. The total number of objective function evaluations for the hybrid scheme is 1689.

For this test example, the best design and values of the constraints obtained by the PSO-SQP method are given in Tables 7.11 and 7.12, respectively. It can be seen that for the above optimum design, all constraints have been met and are active. The minus symbol (-) as a superscript after the constraint value of Table 7.12 and in other tables means that the value of the constraint is slightly less than the one written, but has been rounded. This means that the optimum reaches the feasibility boundary from the "safe" or feasible side, as opposed to the case of a plus symbol (+) which means that the feasibility boundary is reached from the "unsafe" or infeasible side.

Figure 7.21 gives a graphical representation of the best design obtained by the PSO-SQP method, where the thickness of each member is proportional to the corresponding value of each design variable.

Table 7.11 PSO - 10 bar plane truss: Optimum design obtained by the hybrid PSO-SQP.

Property	Value
A1 (in^2)	30.5218
A2 (in^2)	0.1000
A3 (in^2)	23.1999
A4 (in^2)	15.2229
A5 (in^2)	0.1000
A6 (in^2)	0.5514
A7 (in^2)	7.4572
A8 (in^2)	21.0364
A9 (in^2)	21.5285
A10 (in^2)	0.1000
Weight (lb)	5060.85

Table 7.12 PSO - 10 bar plane truss:
Feasibility of the optimum design obtained by the hybrid PSO-SQP.

Constraints	Value	Allowable value	Active
Max Stress (ksi)	25.0000^-	25	●
Max Displacement (in)	2.0000^-	2	●

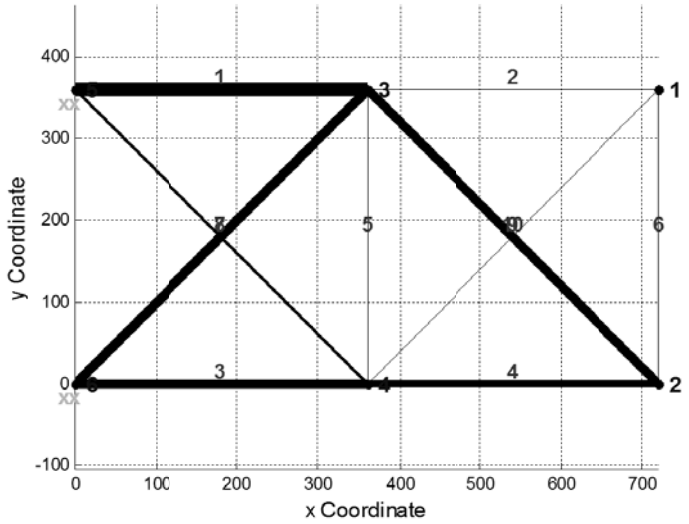


Figure 7.21 PSO - 10 bar plane truss: Graphical representation of optimum design obtained by the hybrid PSO-SQP.

Comparison with results from the literature

For this specific benchmark problem, various results from the literature can be found in (Perez and Behdinan 2007a). In Tables 7.13 and 7.14, the objective function value and the constraints values are calculated for every proposed optimum design. Violated constraints are marked in bold.

Table 7.13 PSO - 10 bar plane truss: Optimum designs from the literature (a).

Property	(Rizzi 1976)	(Gellatly and Berke 1971)	(Schmit and Miura 1976)	(Ghasemi et al. 1997)	(Schmit and Farshi 1974)	(Dobbs and Nelson 1976)	The proposed PSO
A1 (in^2)	30.7300	31.3500	30.5700	25.7300	33.4300	30.5000	30.9810
A2 (in^2)	0.1000	0.1000	0.3690	0.1090	0.1000	0.1000	0.1000
A3 (in^2)	23.9340	20.0300	23.9700	24.8500	24.2600	23.2900	23.1714
A4 (in^2)	14.7330	15.6000	14.7300	16.3500	14.2600	15.4300	15.6935
A5 (in^2)	0.1000	0.1400	0.1000	0.1060	0.1000	0.1000	0.1000
A6 (in^2)	0.1000	0.2400	0.3640	0.1090	0.1000	0.2100	0.5848
A7 (in^2)	8.5420	8.3500	8.5470	8.7000	8.3880	7.6490	7.4298
A8 (in^2)	20.9540	22.2100	21.1100	21.4100	20.7400	20.9800	20.6310
A9 (in^2)	21.8360	22.0600	20.7700	22.3000	19.6900	21.8200	21.3287
A10 (in^2)	0.1000	0.1000	0.3200	0.1220	0.1000	0.1000	0.1000
Weight (lb)	5127.58	5112.62	5107.32	5095.64	5091.50	5080.21	5062.30
Max Stress (ksi)	20.3549	22.9369	20.3959	18.5255	21.1915	24.0675	24.9745
Max Displ. (in)	1.9823	1.99999	1.99998	**2.0137**	1.9998	1.9999	1.999998

Table 7.14 PSO - 10 bar plane truss: Optimum designs from the literature (b).

Property	(Haug and Arora 1979)	(Haftka and Gürdal 1992)	The proposed PSO-SQP	(Adeli and Kamal 1991)	(Perez and Behdinan 2007b)	(El-Sayed and Jang 1994)	(Galante 1992)	(Memari and Fuladgar 1994)
A1 (in^2)	30.0300	30.5200	30.5218	31.2800	33.5000	32.9700	30.4400	30.5610
A2 (in^2)	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
A3 (in^2)	23.2740	23.2000	23.1999	24.6500	22.7660	22.7990	21.7900	27.9460
A4 (in^2)	15.2860	15.2200	15.2229	15.3900	14.4170	14.1460	14.2600	13.6190
A5 (in^2)	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
A6 (in^2)	0.5570	0.5510	0.5514	0.1000	0.1000	0.7390	0.4510	0.1000
A7 (in^2)	7.4680	7.4570	7.4572	7.9000	7.5340	6.3810	7.6280	7.9070
A8 (in^2)	21.1980	21.0400	21.0364	21.5300	20.4670	20.9120	21.6300	19.3450
A9 (in^2)	21.6180	21.5300	21.5285	19.0700	20.3920	20.9780	21.3600	19.2730
A10 (in^2)	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
Weight (lb)	5061.63	5060.93	5060.85	5052.63	5024.25	5013.39	4999.22	4981.06
Max Stress (ksi)	24.9206	**25.0027**	25.0000	23.0690	**25.0171**	**31.2885**	**25.0867**	20.5999
Max Displ. (in)	**2.000004**	1.99996	2.0000	**2.0195**	**2.0389**	**2.0131**	**2.0280**	**2.0605**

It is clear from Tables 7.13 and 7.14 that the best feasible optimum designs are the ones found with the proposed hybrid PSO-SQP and PSO algorithms, since any better design in terms of objective function value violated at least one of the problem constraints.

7.2.2 25 bar space truss

The second test example is a 25-member space truss. The structure is depicted in Figures 7.22 and 7.23. The "xxx" sign for nodes 7, 8, 9, 10 of the model denotes a restriction for all three DOFs of the node (pinned constraint). Variations of this test example can be found in the literature (Perez and Behdinan 2007a; Zhou and Rozvany 1993). The problem described below is the one described in (Zhou and Rozvany 1993), as in (Perez and Behdinan 2007a) the load cases and the stress constraints are different, leading to different, non-comparable results. The nodal coordinates are shown in Table 7.15.

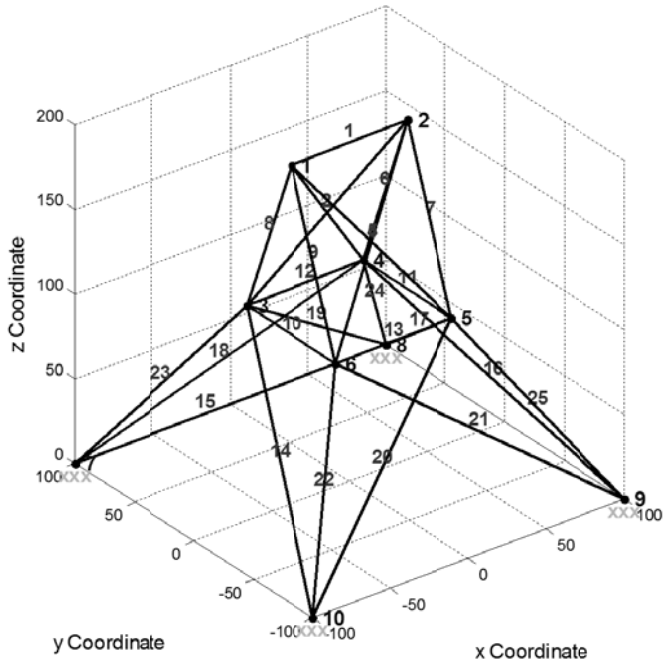


Figure 7.22 PSO - 25 bar space truss: 3D view of the truss model (coordinates in inches).

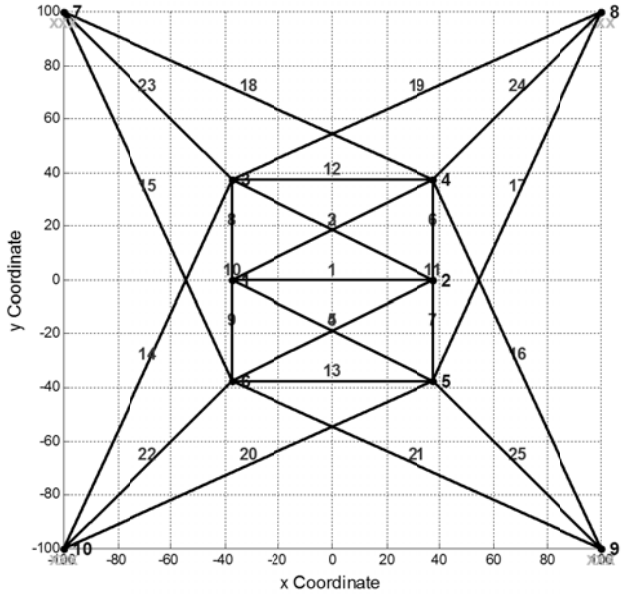


Figure 7.23 PSO - 25 bar space truss: Top view of the truss model (coordinates in inches).

Table 7.15 PSO - 25 bar space truss: Nodal coordinates.

Node	*x* (in)	*y* (in)	*z* (in)
1	-37.5	0	200
2	37.5	0	200
3	-37.5	37.5	100
4	37.5	37.5	100
5	37.5	-37.5	100
6	-37.5	-37.5	100
7	-100	100	0
8	100	100	0
9	100	-100	0
10	-100	-100	0

The structural characteristics are the following: modulus of elasticity E=100000 ksi, material weight ρ=0.1lb/in^3. The structural members are divided into 8 groups. The design variables are the cross section areas of each member group in the range [0.01, 5] (in^2). The 8 design variable groups together with the constraints imposed on stresses for each group are presented in Table 7.16.

Table 7.16 PSO - 25 bar space truss: Design variable groups and allowable stresses.

Design variable	Member	Allowable tension (ksi)	Allowable compression (ksi)
1	1	40	-35.092
2	2-5	40	-11.590
3	6-9	40	-17.305
4	10,11	40	-35.092
5	12,13	40	-35.092
6	14-17	40	-6.759
7	18-21	40	-6.759
8	22-25	40	-11.082

The maximum allowable displacement in the $\pm x$, $\pm y$ and $\pm z$ directions for each node is $d_{\max}$=0.35in. Two load cases have been considered. The nodal loads for each load case are presented in Tables 7.17 and 7.18. The objective to minimize is the weight of the structure under the constraints described above for both load cases simultaneously.

Table 7.17 PSO - 25 bar space truss: Nodal loads – First load case.

Node	F_x (kip)	F_y (kip)	F_z (kip)
1	1	10	-5
2		10	-5
3	0.5		
6	0.5		

Table 7.18 PSO - 25 bar space truss: Nodal loads – Second load case.

Node	F_x (kip)	F_y (kip)	F_z (kip)
1		20	-5
2		-20	-5

The influence of the inertia update rule on the optimization process of the PSO schemes will be investigated first. Three PSO schemes are considered, as in the previous example. The basic PSO parameters used are shown in Table 7.19.

Table 7.19 PSO - 25 bar space truss: PSO parameters used for the inertia update rule check.

Symbol	Value
NP	15
n	8
w	$w_{max}=0.95$ $w_{min}=0.5$
$\boldsymbol{x}^L, \boldsymbol{x}^U$	$x^L{}_i=0.01$, $x^U{}_i=5$ for all dimensions (in^2)
$\boldsymbol{v}^{max}$	$v^{max}{}_i=2.5$ for all dimensions (in^2)
c_1, c_2	$c_1=c_2=2$
t_{max}	200

Since two load cases are considered, the number of finite element analyses needed for each iteration is $2 \cdot NP=30$, while the number of objective function evaluations for each iteration is NP. Ten PSO optimization runs are performed for each of the three cases. The results obtained for 200 PSO iterations are reported in Table 7.20 which shows the objective function values obtained for the best run, the worst run and the average of the 10 runs for each case.

Table 7.20 PSO - 25 bar space truss: Statistical results for 10 PSO runs after 200 iterations.

Result	Fixed rule ($w=w_{max}$)	Linear rule	Non-linear cubic rule ($a_w=1.3$)
Best (lb)	599.79	547.78	545.45
Worst (lb)	679.46	604.92	558.97
Average (lb)	627.35	557.14	549.08

Figure 7.24 depicts the convergence history for the three cases, in terms of the average result over 10 optimization runs. Furthermore, the performance of the algorithm is studied with the convergence criterion $f_m=10^{-6}$ connected to the relative improvement of the objective function over the last $k_f=30$ iterations. The results reported in Table 7.21 include the average number of iterations needed for convergence and the objective function values obtained for the best run, the worst run and the average of the 10 runs. The non-linear rule outperforms the other two rules in terms of the best result, the worst result and the average result for 10 runs, due to its smoother convergence characteristics.

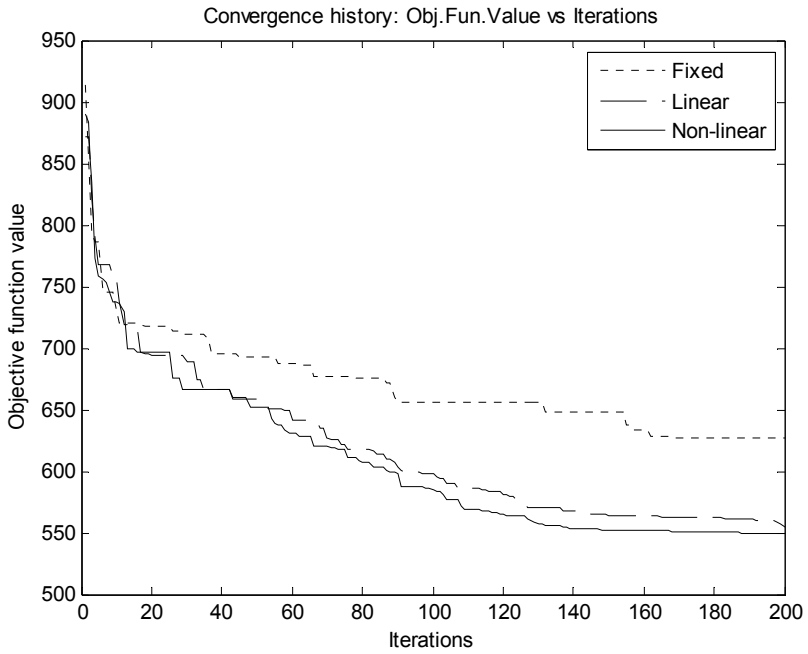


Figure 7.24 PSO - 25 bar space truss: Convergence history for the three PSO schemes.

Table 7.21 PSO - 25 bar space truss: Statistical results for 10 PSO runs.

Result	Fixed rule ($w = w_{max}$)	Linear rule	Non-linear rule ($\alpha_w = 1.3$)
Average number of iterations	107	144	175
Best (lb)	581.72	576.80	546.12
Worst (lb)	697.88	692.44	694.15
Average (lb)	659.60	633.97	576.16

Comparison of PSO with Evolution Strategies

In order to investigate the performance of the proposed PSO algorithm with respect to established Evolutionary Programming algorithms, we consider one of the most efficient optimization algorithms, namely the Evolution Strategies (ES), for comparison. The ES version implemented is a (10+15) ES version with 10 parents and 15 offspring and the latest version of a series of improvements developed by Papadrakakis et al. (2001b). The PSO scheme is implemented with 15 individuals. Ten independent ES runs are also performed. For both methods, 30 finite element analyses are needed for every iteration (or generation).

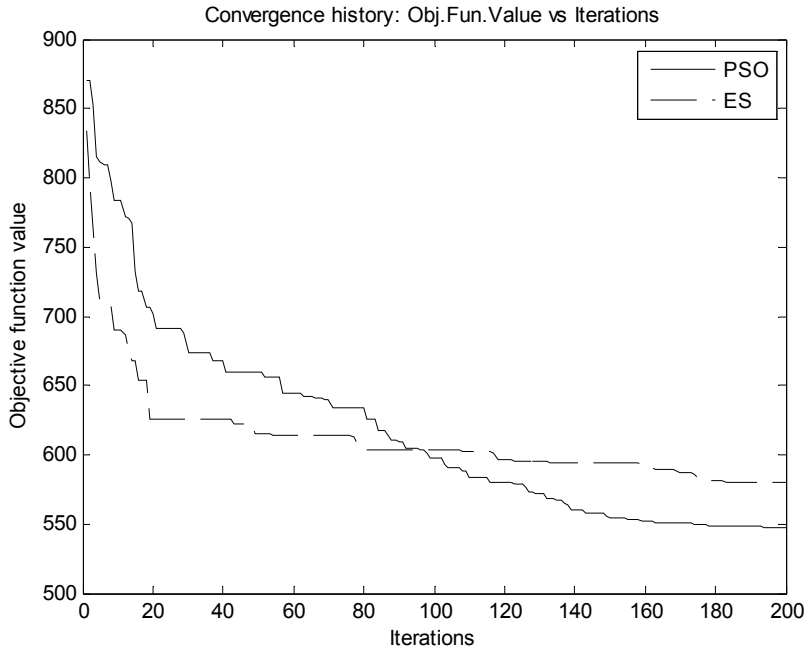


Figure 7.25 PSO - 25 bar space truss:
Convergence history for PSO and ES (average of ten runs).

A given number of iterations (200) is adopted as the termination criterion. Figure 7.25 shows the convergence history of the two optimization methods, where the horizontal axis represents the PSO iterations or the ES generations. It can be seen that the ES per-

forms better than the PSO during the first 95 iterations, however its convergence rate deteriorates substantially after that point. On the contrary, PSO continues to converge in a uniform rate until it reaches the lowest value of 548.30 (in average) for the objective function.

A combination of ES and PSO

Taking into consideration the results of the previous study, where the ES was shown to perform better than the PSO during the first iterations while the PSO performed better at the end of the optimization process, we investigated a combined ES-PSO scheme, where ES is implemented at the beginning and PSO is implemented after the ES has reached a plateau of no substantial reduction of the objective function for a certain number of generations. The PSO takes over after 95 ES generations with initial conditions the final generation of the ES. All members of the swarm are initialized at the position of the best estimate of the ES and they are given random velocities. Figure 7.26 shows the convergence history for the hybrid ES-PSO scheme compared to the ES and PSO schemes. A similar test was conducted with the PSO taking over after 40 ES generations and the results are depicted in Figure 7.27.

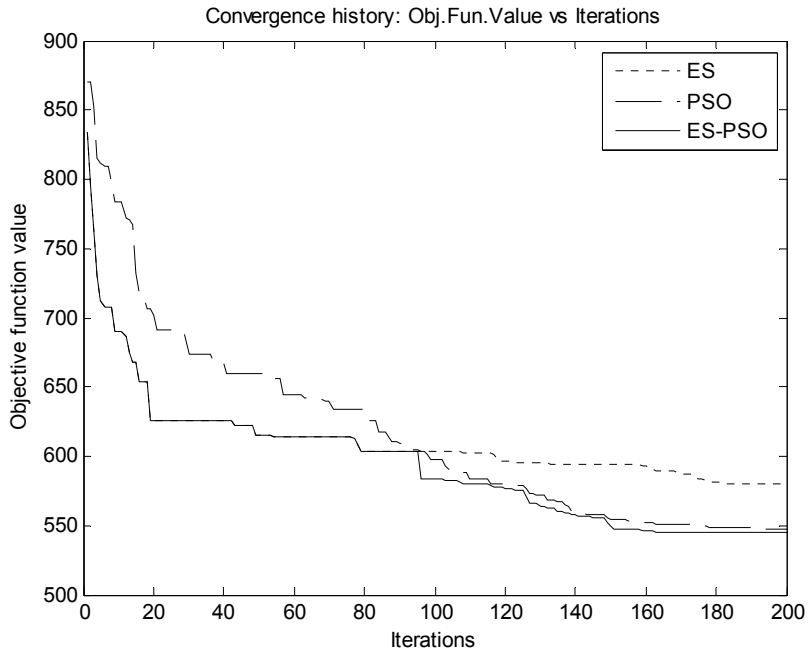


Figure 7.26 PSO - 25 bar space truss: Convergence history for the combined ES-PSO (a).

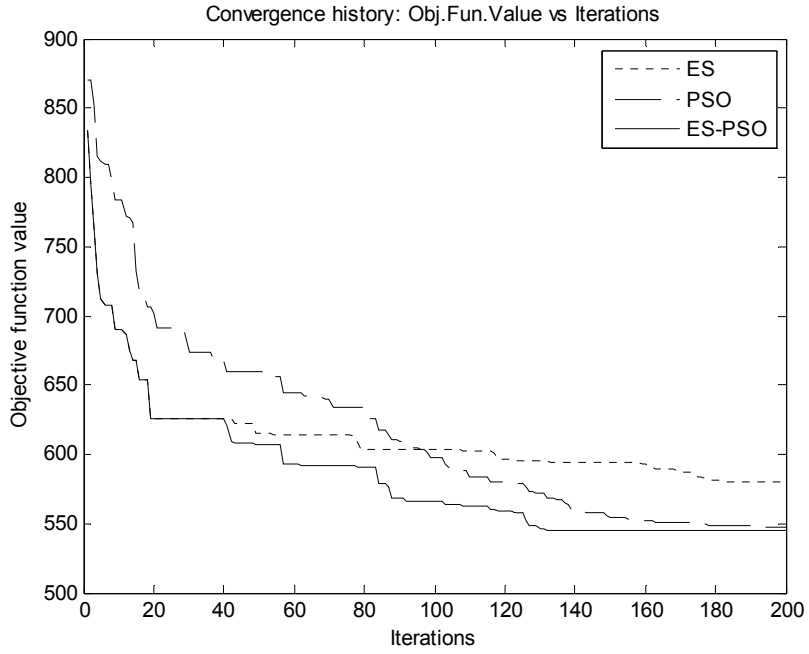


Figure 7.27 PSO - 25 bar space truss: Convergence history for the combined ES-PSO (b).

Next we will examine a combined ES-PSO scheme using a termination criterion for the ES based on the improvement of the objective function. If the relative improvement of the objective function over the last k_f=30 ES generations is less or equal to f_m=10^{-6}, then the PSO starts from the best estimate of the ES, given random velocities. Figure 7.28 shows the history of the combined scheme compared to the previous ES and PSO schemes, where the horizontal axis represents the PSO iterations or the ES generations. It can be seen that the combined method produces the same quality of the final result as the PSO method, while its convergence rate is better over the iterations, while the difference between the ES and ES-PSO schemes until generation 86 is due to the stochastic nature of the ES method.

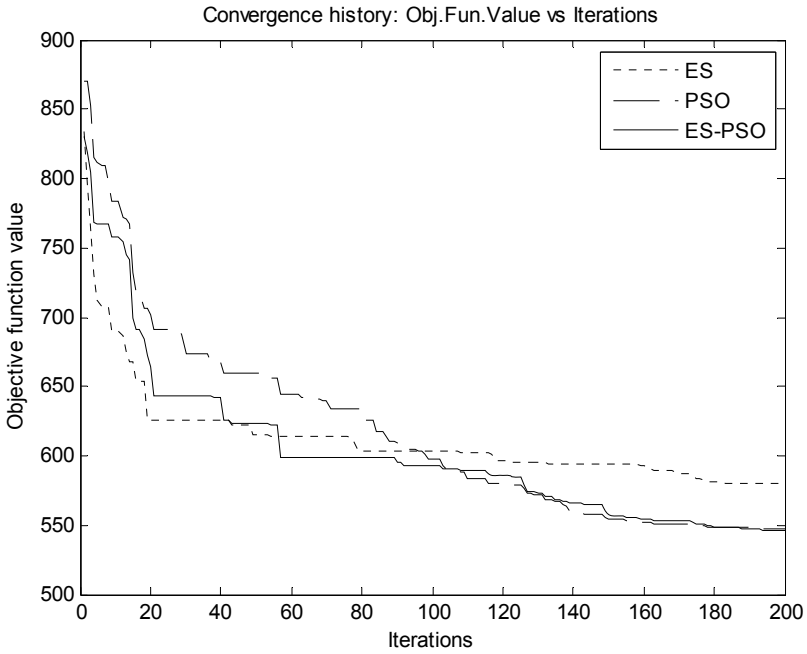


Figure 7.28 PSO - 25 bar space truss: Convergence history for the combined ES-PSO (c).

The hybrid PSO-SQP method

First we apply the SQP alone, for various initial designs. Four initial designs have been selected, namely the ones corresponding to design variables values 5, 3.5, 2 and 0.5 for every dimension of the problem. It can be seen that the SQP method converges to sub-optimal design points, or does not converge at all, as a good model initialization is always required for SQP to produce good results. Next, we implement the PSO-SQP scheme. The termination criterion and other settings are the same as those used for the previous PSO-SQP test example.

Table 7.22 PSO - 25 bar space truss: Convergence behavior of SQP.

Starting point (in^2)	Iterations	Obj. fun. evaluations	Obj. function value
"5"	34	396	825.991
"3.5"	41	514	825.991
"2"	18	308	653.357
"0.5"	100	1204	No convergence

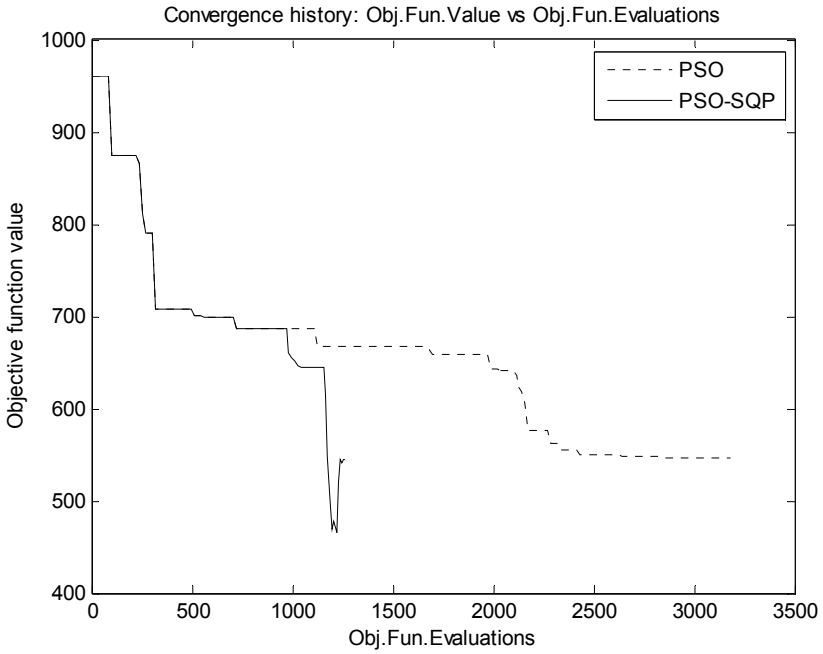


Figure 7.29 PSO - 25 bar space truss: Convergence history for the hybrid PSO-SQP.

The convergence history of the hybrid PSO-SQP scheme, compared to the simple PSO scheme is depicted in Figure 7.29 in terms of objective function value and the objective function evaluations. In the hybrid scheme, the PSO needs 960 function evaluations to reach an objective function value of 687.097, while from that point on, SQP needs 306 objective function evaluations to converge to an optimum value of 545.037. The total number of objective function evaluations for the hybrid scheme is 1266.

The best design obtained by the hybrid PSO-SQP method, together with the best result of the PSO employing the non-linear rule and the results of Zhou & Rozvany (Zhou and Rozvany 1993) are presented in Table 7.23.

Table 7.23 PSO - 25 bar space truss: PSO results.

Property	(Zhou and Rozvany 1993)	PSO (This study)	Hybrid PSO-SQP (This study)
A1 (in^2)	0.0100	0.01000	0.01000
A2 (in^2)	1.9870	2.0363	2.04300
A3 (in^2)	2.9935	3.1216	3.00239
A4 (in^2)	0.0100	0.01000	0.01000
A5 (in^2)	0.0100	0.01000	0.01000
A6 (in^2)	0.6840	0.6740	0.68337
A7 (in^2)	1.6769	1.5771	1.62296
A8 (in^2)	2.6621	2.6657	2.67194
Weight (lb)	545.163	545.45	545.037

It can be seen that the best design, in terms of objective function value, is the one achieved by PSO-SQP. It can also be seen that although the result of Zhou & Rozvany is slightly better than the one obtained with the proposed PSO, there is a slight violation of the maximum nodal displacement constraint. For the optimum designs achieved by PSO and PSO-SQP, all constraints have been met while the maximum nodal displacement constraint is active. Figure 7.30 gives a graphical representation of the best design obtained by the PSO-SQP method, where the thickness of each member is proportional to the corresponding value of each design variable.

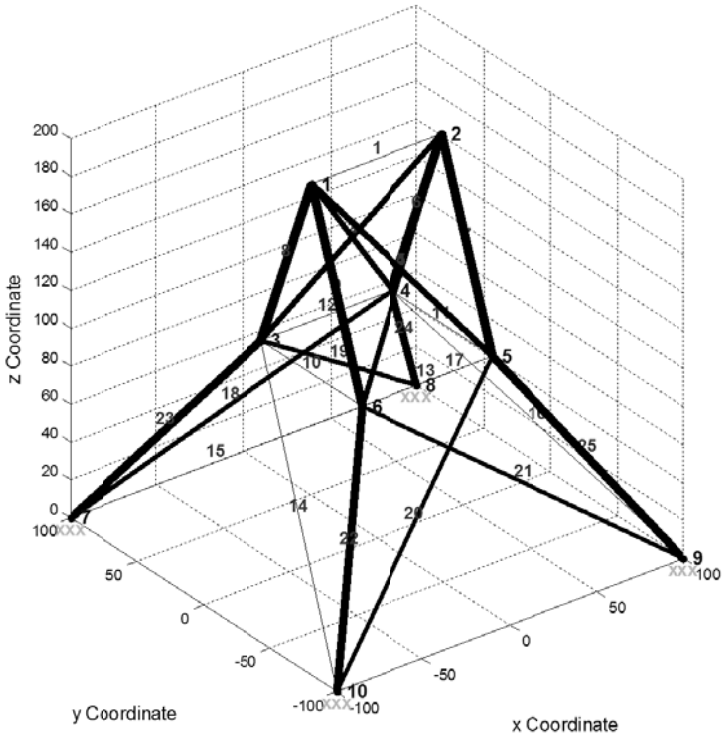


Figure 7.30 PSO - 25 bar space truss: Graphical representation of the optimum design obtained by the hybrid PSO-SQP.

Table 7.24 PSO - 25 bar space truss: Feasibility of the optimum design.

		(Zhou and Rozvany 1993)		PSO (This study)		Hybrid PSO-SQP (This study)	
Constraints	**Allowable value**	**Value**	**Active**	**Value**	**Active**	**Value**	**Active**
Max stress (ksi)	40	6.9846	○	7.3735	○	7.1580	○
Max Nodal Displ. (in)	0.35	**0.3500012**	●	0.3499	●	0.3500	●
Group 1, 4, 5 Min stress (ksi)	-35.092	-5.2989	○	-5.3741	○	-5.4084	○
Group 2 Min stress (ksi)	-11.590	-6.9382	○	6.8239	○	-6.8192	○
Group 3 Min stress (ksi)	-17.305	-4.8531	○	-4.6253	○	-4.7991	○
Group 6, 7 Min stress (ksi)	-6.759	-5.3200	○	-5.6338	○	-5.4665	○
Group 8 Min stress (ksi)	-11.082	-4.0980	○	-4.1084	○	-4.1037	○

7.2.3 72 bar space truss

The third test example is a space truss with 72 members, shown in Figures 7.31 and 7.32. It can be found in the work of Adeli and Kamal (1986) , Adeli and Park (1998), Sarma and Adeli (2000), among others. The modulus of Elasticity is E=10000ksi and the material weight ρ=0.1lb/in^3. The basis of the structure is a rectangle with a side of 120 in, while the total height is 4×60in=240in. The "xxx" sign for the basis nodes of the model denotes a restriction for all three DOFs of the node (pinned constraint). The nodal coordinates are shown in Table 7.25. The basis nodes, 1, 2, 3, 4 are fixed on the ground. The structural members are divided into 16 groups, shown in Table 7.26. Two load cases as considered. The nodal loads for each load case are presented in Tables 7.27 and 7.28. The element connectivity for elements 1-18 of the first floor of the structure is shown in Figure 7.33. For the other elements 19-72 of floors 2-4, the connectivity has the same pattern as the one of the first floor. The design variables are the cross section areas of each member group with a lower limit of 0.01in^2 and no upper limit. The constraints are imposed on stresses and displacements. The maximum allowable displacement in the $\pm x$ and $\pm y$ directions for each node is $d_{\max}$=0.25in, while the maximum allowable stress (absolute value) is σ_{allow}=25ksi in tension or compression and the objective is to minimize the weight of the structure under the specified constraints.

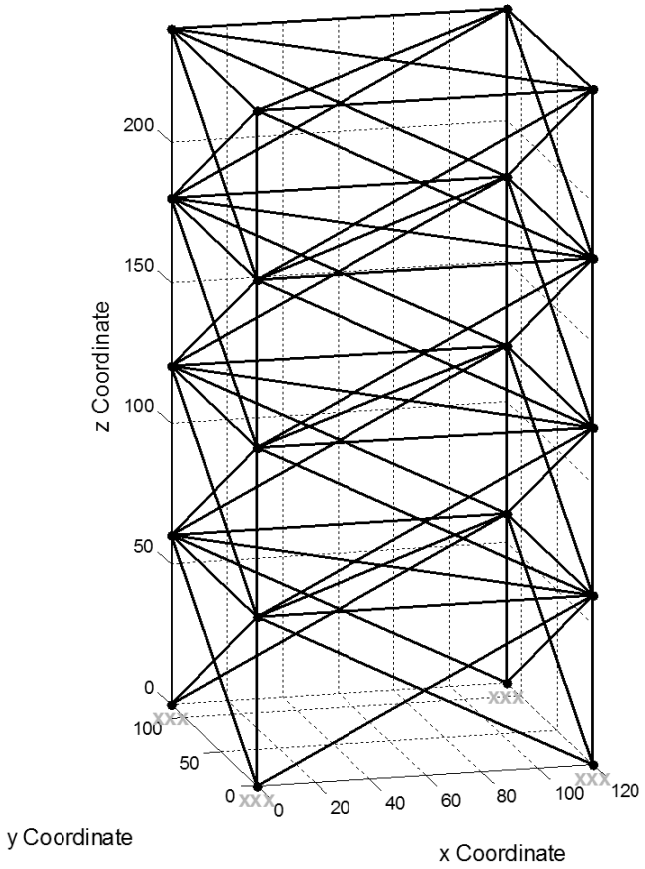


Figure 7.31 PSO - 72 bar space truss: 3D view of the model (coordinates in inches).

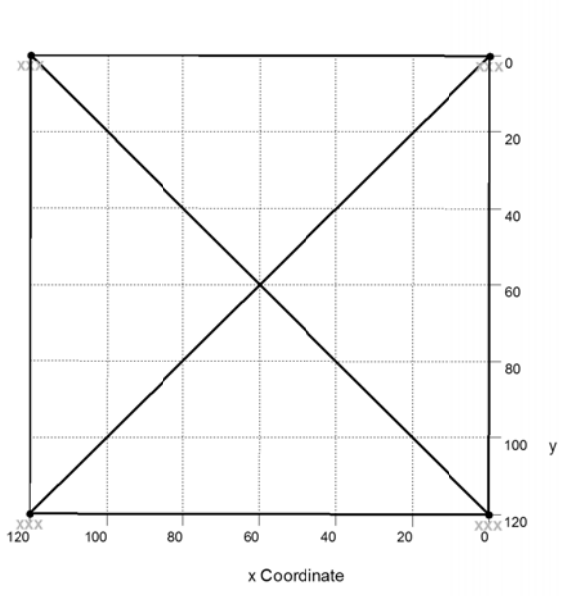


Figure 7.32 PSO - 72 bar space truss: Top view of the model (coordinates in inches).

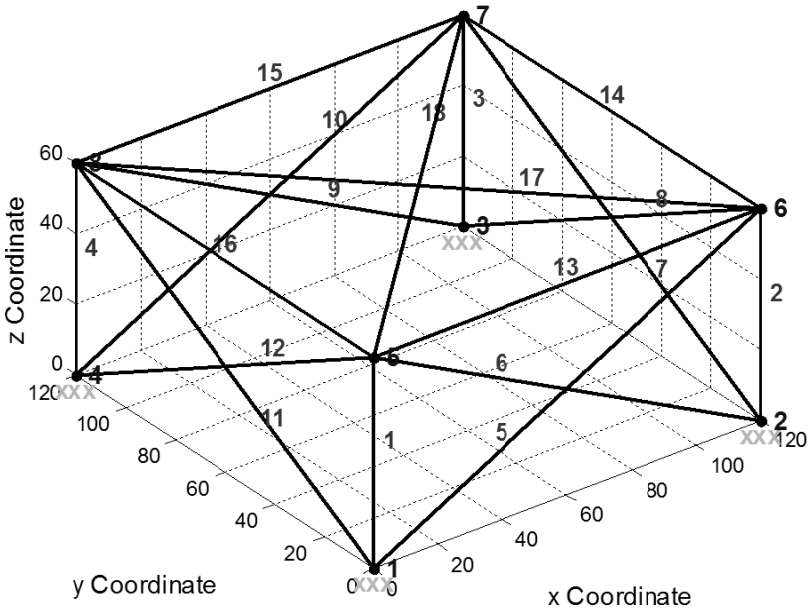


Figure 7.33 PSO - 72 bar space truss: Element connectivity for the first floor.

Table 7.25 PSO - 72 bar plane truss: Nodal coordinates (in inches).

Node	*x* (in)	*y* (in)	*z* (in)	Node	*x* (in)	*y* (in)	*z* (in)
1	0	0	0	11	120	120	120
2	120	0	0	12	0	120	120
3	120	120	0	13	0	0	180
4	0	120	0	14	120	0	180
5	0	0	60	15	120	120	180
6	120	0	60	16	0	120	180
7	120	120	60	17	0	0	240
8	0	120	60	18	120	0	240
9	0	0	120	19	120	120	240
10	120	0	120	20	0	120	240

Table 7.26 PSO - 72 bar plane truss: Design variable groups.

Design variable	Member	Design variable	Member
A1	1-4	A9	37-40
A2	5-12	A10	41-48
A3	13-16	A11	49-52
A4	17,18	A12	53,54
A5	19-22	A13	55-58
A6	23-30	A14	59-66
A7	31-34	A15	67-70
A8	35,36	A16	71,72

Table 7.27 PSO - 72 bar plane truss: Nodal loads – First load case.

Node	F_x (kip)	F_y (kip)	F_z (kip)
17	5	5	-5

Table 7.28 PSO - 72 bar plane truss: Nodal loads – Second load case.

Node	F_x (kip)	F_y (kip)	F_z (kip)
17	0	0	-5
18	0	0	-5
19	0	0	-5
20	0	0	-5

Comparison of PSO with GA

This test example is considered in order to compare the proposed PSO methodology with Genetic Algorithms. At first, the PSO algorithm itself is implemented with the convergence criterion $f_m=10^{-6}$ connected to the relative improvement of the objective function over the last $k_f=150$ iterations. The objective is to compare the convergence properties of the proposed PSO methodology with the corresponding GA results of Sarma and Adeli (2000). The characteristics of the PSO scheme used are shown in Table 7.29.

Table 7.29 PSO - 72 bar plane truss: PSO parameters used.

Symbol	Value
NP	40
n	16
w	$w_{max} = 0.95$ $w_{min} = 0.5$
$\boldsymbol{x}^L, \boldsymbol{x}^U$	$x^L{}_i = 0.01$ for all dimensions (in^2)
$\boldsymbol{v}^{max}$	$v^{max}{}_i = 1.5$ for all dimensions (in^2)
c_1, c_2	$c_1 = c_2 = 2$
t_{max}	2000
α_w	1.3

The convergence history of PSO is shown in Figure 7.34. The optimum design achieved is 364.01 lb, obtained in 1512 iterations. The optimum design achieved is shown in Table 7.30 and compared with the results obtained by Adeli and Park (1998) and Sarma and Adeli (2000). It is shown that the proposed PSO algorithm converged to a slightly better value of the objective function, in less iterations than the two GA algorithms.

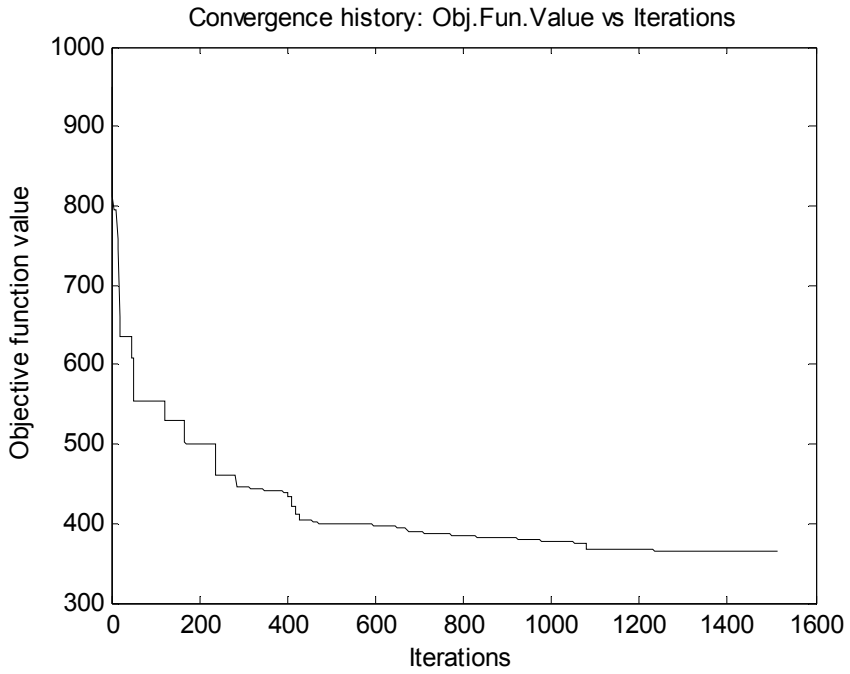


Figure 7.34 PSO - 72 bar space truss: Convergence history for the PSO.

Table 7.30 PSO - 72 bar plane truss: Comparison of the optimum design with results from the literature.

Property	(Adeli and Park 1998)	(Sarma and Adeli 2000) Simple GA	(Sarma and Adeli 2000) Fuzzy GA	The proposed PSO	The proposed PSO-SQP
A1 (in^2)	2.7547	2.1407	1.7321	1.8497	1.8875
A2 (in^2)	0.5102	0.5098	0.5215	0.5217	0.5169
A3 (in^2)	0.0100	0.0538	0.0100	0.0100	0.0100
A4 (in^2)	0.0100	0.0100	0.0129	0.0101	0.0100
A5 (in^2)	1.3696	1.4889	1.3451	1.3041	1.2901
A6 (in^2)	0.5070	0.5507	0.5507	0.5225	0.5170
A7 (in^2)	0.0100	0.0568	0.0100	0.0100	0.0100
A8 (in^2)	0.0100	0.0129	0.0129	0.0100	0.0100
A9 (in^2)	0.4807	0.5653	0.4923	0.5215	0.5211
A10 (in^2)	0.5084	0.5273	0.5449	0.5014	0.5181
A11 (in^2)	0.0100	0.0100	0.0655	0.0119	0.0100
A12 (in^2)	0.0643	0.0655	0.0129	0.1257	0.1140
A13 (in^2)	0.2151	0.1737	0.1778	0.1651	0.1665
A14 (in^2)	0.5179	0.4250	0.5244	0.5442	0.5362
A15 (in^2)	0.4190	0.4367	0.3958	0.4465	0.4457
A16 (in^2)	0.5039	0.6413	0.5952	0.5783	0.5759
Weight (lb)	376.50	372.40	364.40	364.01	363.82
Iterations	-	2776	1758	1512	-

Furthermore, it can be seen in Table 7.31 that the optimum design of the proposed PSO methodology is truly feasible, confirming that the proposed method yields always feasible optimum designs. Violated constraints are highlighted in bold.

Table 7.31 PSO - 72 bar plane truss: Comparison of the constraints of the optimum design with results from the literature.

Constraints	Allowable value	(Adeli and Park 1998)	(Sarma and Adeli 2000) Simple GA	(Sarma and Adeli 2000) Fuzzy GA	The proposed PSO	The proposed PSO-SQP
Max Abs. *x*-displ. (in)	0.25	0.2494	0.2500^{+}	**0.2523**	0.2500^{-}	0.2500^{+}
Max Abs. *y*- displ. (in)	0.25	0.2494	0.2500^{+}	**0.2523**	0.2500^{-}	0.2500^{+}
Min. stress (ksi)	-25	-6.8170	-7.2150	-7.2885	-6.9616	-7.1494
Max. stress (ksi)	25	20.7658	24.8790	24.0550	24.9759	25.0000^{+}

The hybrid PSO-SQP method

Next, the hybrid PSO-SQP scheme is applied. If the relative improvement of the objective function over the last k_f=95 iterations of the PSO optimizer is less or equal to f_m=10^{-5}, the PSO phase is terminated and subsequently the SQP starts from the best estimate of the PSO. The SQP termination criterion is the same as the one used for the

previous test example. The results of the hybrid method are also reported in Tables 7.30 and 7.31. It is shown that the optimum result achieved by the PSO-SQP is slightly better than that of PSO, without violating the constraints. The main advantage of the PSO-SQP method is its fast convergence rate, as shown in Figure 7.35, where the convergence history of the hybrid PSO-SQP scheme is depicted in terms of objective function value vs. objective function evaluations.

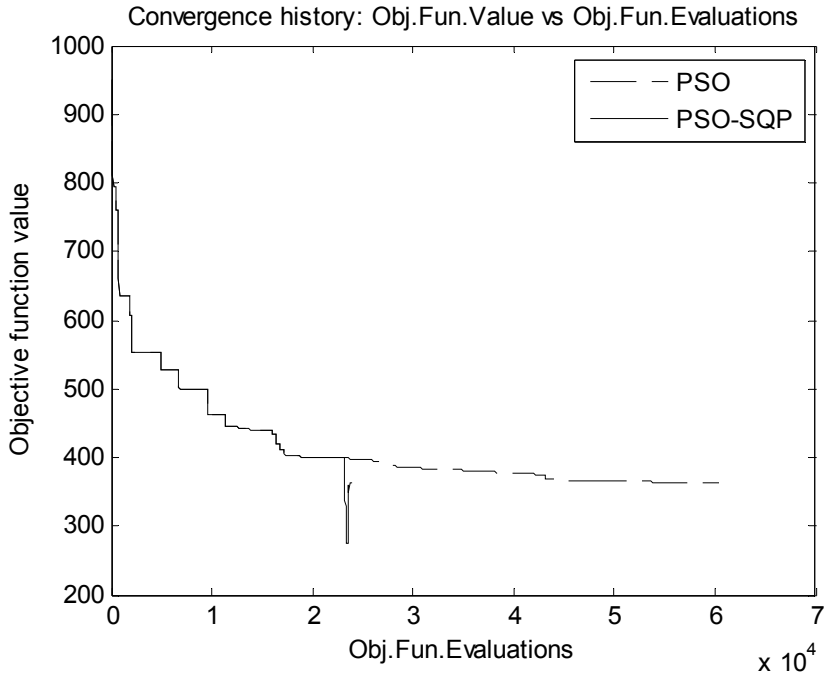


Figure 7.35 PSO - 72 bar space truss: Convergence history for the hybrid PSO-SQP.

In the hybrid scheme, the PSO needs 579 iterations, or 23160 objective function evaluations to reach an objective function value of 399.78, while from that point on, SQP needs 715 additional objective function evaluations to converge to an optimum value of 363.82. The total number of objective function evaluations for the hybrid scheme is 23875 compared to 60480 for the PSO scheme. Figure 7.36 gives a graphical representation of the best design obtained by the PSO-SQP method, where the thickness of each member is proportional to the corresponding value of each design variable.

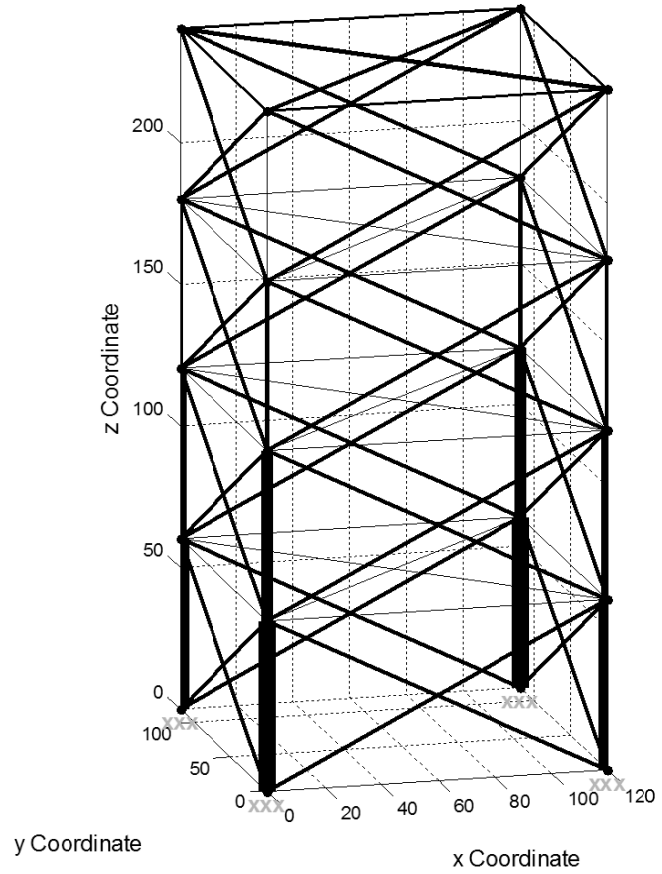


Figure 7.36 PSO - 72 bar space truss: Graphical representation of the optimum design obtained by the hybrid PSO-SQP.

7.2.4 Conclusions on the PSO test examples

An optimization algorithm for the optimum structural design based on the Particle Swarm Optimization algorithm is introduced. A non-linear weight update rule for PSO, an efficient constraint handling technique and a hybrid PSO-SQP scheme for global structural optimization are proposed and evaluated in three benchmark problems. The PSO algorithm showed efficient and robust performance in all the test examples, achieving better results than other evolutionary algorithms such as ES and GAs, in terms of final optimum and convergence speed.

The non-linear weight update rule for PSO showed better performance than the fixed or the linear rule, especially in cases where a termination criterion connected to the relative improvement of the objective function was used, exhibiting smoother convergence. The constraint handling technique used in the study, based on a linear segment penalty function, showed excellent performance, since it always led to feasible optimal designs, while taking also advantage of infeasible designs during the optimization procedure.

The proposed hybrid algorithm based on the PSO and SQP is a well-suited optimization tool for solving non-convex optimization problems, in identifying the global optimum

from multiple local ones. It managed to improve further the PSO results, achieving better or the same quality of final result, in significantly better computational time. The numerical results demonstrated the efficiency of the proposed hybrid PSO-SQP algorithm for structural optimization problems.

In the standard PSO procedure, the characteristic parameters have to be fine-tuned carefully, based on the experience of the designer, or on trial and error and any other information available for the specific problem at hand. The selection of the PSO parameters plays a significant role in the result of the process in terms of both convergence rate and final optimum design achieved. In general, a bad selection of these parameters can lead to a poor result. By using the proposed hybrid PSO-SQP methodology, the significance of the PSO parameters in the performance of the method is substantially alleviated. There is no need of fine-tuning the PSO algorithm for obtaining a high quality final result since the SQP optimization phase can improve drastically the PSO solution and increase significantly the robustness of the optimization scheme.

Chapter 8

8 Numerical Applications – Part B: Probabilistic Optimization

This chapter contains the second part (Part B) of the numerical applications investigated in the thesis, where probabilistic optimization is considered. In this chapter, ten test examples are considered in total. The chapter is divided into five sections:

1. In the first section (Section 8.1), two *Robust Design Optimization* (RDO) test examples are considered, using standard methods for solving the multi-objective optimization problem;
2. In the second section (Section 8.2), two RDO test examples are considered, using the proposed non-dominant CEATm methodology for solving the multi-objective optimization problem;
3. In the third section (Section 8.3), two *Reliability-Based Design Optimization* (RBDO) test examples are considered, assisted by *Neural Network* (NN) predictions;
4. In the fourth section (Section 8.4), two *Reliability-based Robust Design Optimization* (RRDO) test examples are considered;
5. In the fifth section (Section 8.5), two RRDO test examples are considered, assisted by NN predictions.

8.1 Robust Design Optimization (RDO) with standard multi-objective methods

The first part of the probabilistic optimization test examples section includes two Robust Design Optimization test examples, namely a 13-bar plane truss bridge and a 39-bar space truss. For both problems, a multi-objective optimization formulation is used, where the objective is to minimize the weight of the structure and the variance of the structural response.

A discrete ES scheme is used, where the design variables (corresponding to the cross sections of the members of the structures) are taken from Eurocode tables. The multi-objective optimization problem is solved with the *Linear Weighting Method* (LWM) of Section 4.9 where all objectives are combined into a single scalar parameterized objective using weight coefficients, while for dealing with the uncertain parameters the *Monte Carlo Simulation* (MCS) method of Section 2.6 is used. The conclusions for both test examples are given together in Section 8.1.3.

8.1.1 13-bar plane truss bridge – RDO test example

A two dimensional 13-bar truss, shown in Figure 8.1, is considered for presenting the efficiency of the proposed RDO methodology (Papadrakakis et al. 2004a). The truss structure corresponds to the finite element model of the Gateway Bridge over the Fork River, Idaho in the USA, which was built in 1948. Two objective functions are used, the weight and the variance of the response of the structure, under constraints on stresses and displacements imposed by the design codes (CEN 1991; CEN 1993).

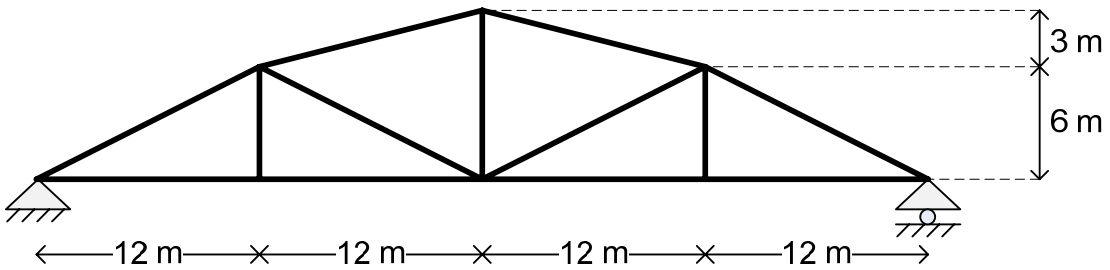


Figure 8.1 RDO - 13-bar plane truss bridge: Model.

Due to engineering practice demands, the members are divided into groups having the same design variables. The design variables are discrete, the dimensions of the members of the structure, four groups in total, taken from the double *Equal Angle Section* (EAS) table of the Eurocode. Table 8.1 shows the single EAS sections of the Eurocode, while Figure 8.2 shows a 3D view of a single EAS section. A double EAS section is composed of two single EAS sections with a distance *d* between them.

For this test example, three types of constraints have been considered, as imposed by the European design code (CEN 1993):

i. Stress constraints;

ii. Compression force constraints (for buckling);

iii. Displacement constraints.

Table 8.1 RDO - 13-bar plane truss bridge: Equal Angle Section of the Eurocode.

Equal Angle Section	Height (mm)	Area (cm^2)	Moment of inertia (cm^4)	Equal Angle Section	Height (mm)	Area (cm^2)	Moment of inertia (cm^4)
20 × 3	20	1.12	0.39	80 × 6	80	9.35	55.8
25 × 3	25	1.42	0.79	80 × 8	80	12.3	72.3
25 × 4	25	1.85	1.01	80 × 10	80	15.1	87.5
30 × 3	30	1.74	1.41	90 × 7	90	12.2	92.6
30 × 4	30	2.27	1.81	90 × 9	90	15.5	116
30 × 5	30	2.78	2.16	100 × 8	100	15.5	145
35 × 4	35	2.67	2.96	100 × 10	100	19.2	177
35 × 5	35	3.28	3.56	100 × 12	100	22.7	207
40 × 4	40	3.08	4.48	110 × 10	110	21.2	239
40 × 5	40	3.79	5.43	120 × 10	120	23.2	313
45 × 4	45	3.49	6.43	120 × 11	120	25.4	341
45 × 5	45	4.3	7.83	120 × 12	120	27.5	368
50 × 5	50	4.8	11	130 × 12	130	30	472
50 × 6	50	5.69	12.8	140 × 13	140	35	638
50 × 7	50	6.56	14.6	150 × 12	150	34.8	737
55 × 6	55	6.31	17.3	150 × 14	150	40.3	845
60 × 5	60	5.82	19.4	150 × 15	150	43	898
60 × 6	60	6.91	22.8	160 × 15	160	46.1	1100
60 × 8	60	9.03	29.1	160 × 17	160	51.8	1230
65 × 7	65	8.7	33.4	180 × 16	180	55.4	1680
70 × 6	70	8.13	36.9	180 × 18	180	61.9	1870
70 × 7	70	9.4	42.4	200 × 16	200	61.8	2340
70 × 9	70	11.9	52.6	200 × 18	200	69.1	2600
75 × 7	75	10.1	52.4	200 × 20	200	76.4	2850
75 × 8	75	11.5	58.9	200 × 24	200	90.6	3330

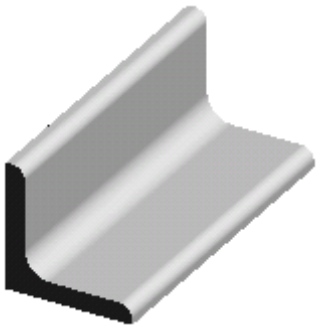


Figure 8.2 RDO - 13-bar plane truss bridge:
3D view of an Equal Angle Section of the Eurocode.

The stress constraint considered can be written as follows

$$\sigma_{\max} \leq \sigma_{\mathrm{a}} \tag{8.1}$$

$$\sigma_{\mathrm{a}} = \frac{\sigma_y}{1.10} \tag{8.2}$$

where $\sigma_{\max}$ is the maximum axial stress (absolute value) in each element group for all loading cases, σ_{a} is the allowable axial stress according to Eurocode 3 (CEN 1993) and σ_y is the yield stress of the material. For members under compression the additional buckling constraint is implemented as

$$\left| P_{\mathrm{c,max}} \right| \leq P_{\mathrm{cc}} \tag{8.3}$$

$$P_{\mathrm{cc}} = \frac{P_{\mathrm{e}}}{1.05} \tag{8.4}$$

$$P_{\mathrm{e}} = \frac{\pi^2 EI}{L_{\mathrm{eff}}^2} \tag{8.5}$$

where $P_{\mathrm{c,max}}$ is the maximum axial compression force for all loading cases, P_e is the critical Euler buckling force in compression, taken as the first buckling mode of a pin-connected member, and L_{eff} is the effective length of the member. The effective length is taken equal to the actual length of the corresponding member. Similarly, the displacement constraint can be written as

$$\left| d \right| \leq d_{\mathrm{a}} \tag{8.6}$$

where d_{a} is the limit value of the displacement at a certain node or the maximum nodal displacement. A constraint of 200mm on the maximum deflection is imposed.

For each design variable, three stochastic variables are assigned: the length L, the width t of the legs and the distance d between the two identical equal angle sections. A vertical load of 300kN is applied to the middle node and 100kN to the rest of the nodes of the deck, both considered as probabilistic actions. The types of probability density functions, the mean values, and the variances of the random parameters are presented in Table 8.2. For this test case the $(\mu+\lambda)$-ES approach is used with $\mu=\lambda=5$, while a sample size of 1000 simulations is taken for the MCS needed for the stochastic analysis. The multi-objective optimization problem is solved with the Linear Weighting Method, where all objectives are combined into a single scalar parameterized objective using weight coefficients, as described in Section 4.7.1.

Table 8.2 RDO - 13-bar plane truss bridge: Characteristics of the random variables.

Symbol (units)	Description	Probability Density Function	Mean value μ	Standard Deviation σ	σ/μ	95% of values within
E (kN/m^2)	Young's Modulus	Normal	2.10×10^8	1.50×10^7	7.14%	(1.81×10^8, 2.39×10^8)
σ_y(kN/m^2)	Allowable stress	Normal	355000	35500	10.00%	(2.85×10^5, 4.25×10^5)
V (kN)	Vertical loading	Normal	V_μ	$10\cdot V_\mu$	10.00%	($2.85\cdot V_\mu$, $4.25\cdot V_\mu$)
L	Legs length	Normal	L_i *	$0.02\cdot L_i$	2%	($0.961\cdot L_i$, $1.039\cdot L_i$)
t	Legs width	Normal	t_i *	$0.02\cdot t_i$	2%	($0.961\cdot t_i$, $1.039\cdot t_i$)
d	EAS section Distance	Normal	d_i *	$0.02\cdot d_i$	2%	($0.961\cdot d_i$, $1.039\cdot d_i$)

* Taken from the double Equal Angle Section table of the Eurocode for every design.

The resultant Pareto front curve is depicted in Figure 8.3, with the weight of the structure on the horizontal and the standard deviation of the horizontal displacement on the vertical axis.

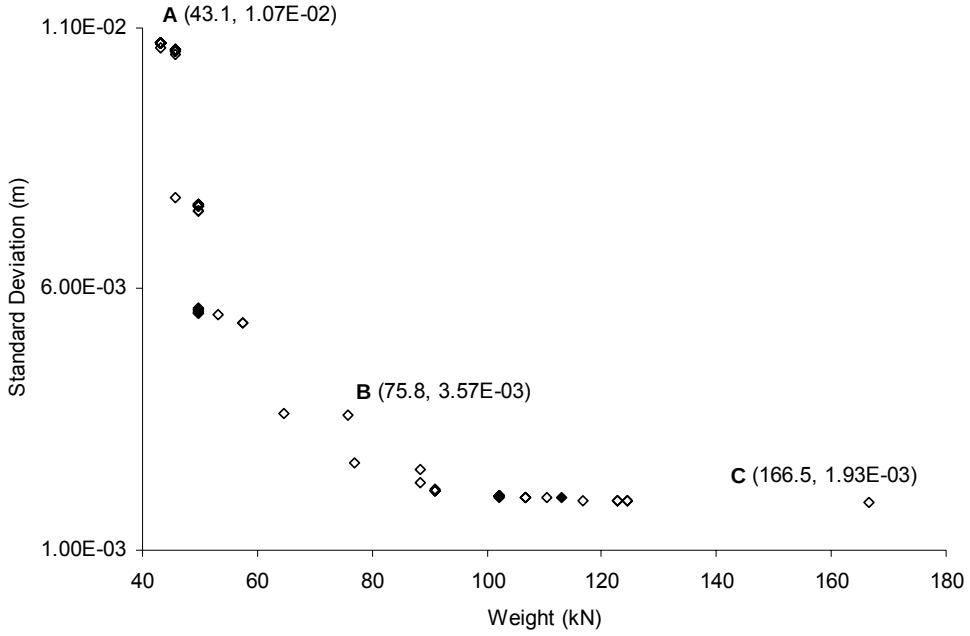


Figure 8.3 RDO - 13-bar plane truss bridge: Pareto Front curve.

From the Pareto front curve, the difference between *Deterministic Design Optimization* (DDO) and RDO optimum designs is demonstrated in terms of the structural weight, the variance of the response and the probability of violation of the constraints. The two ends of the Pareto front curve represent two extreme designs. Point *A* corresponds to the de-

terministic optimum where the weight of the structure is the dominant criterion. Point *C* is the optimum when the standard deviation of the response is considered as the dominant criterion. The intermediate Pareto optimal solutions (such as the one corresponding to point *B*) are compromise solutions between these two extreme optimum designs under conflicting criteria.

8.1.2 39-bar space truss – RDO test example

A three dimensional 39-bar truss, shown in Figure 8.4, is considered for testing the efficiency of the proposed RDO methodology (Papadrakakis et al. 2004b; Papadrakakis et al. 2005; Papadrakakis et al. 2004c). The height of the structure is 16 m, while its basis is an equilateral triangle of side 6.93m.

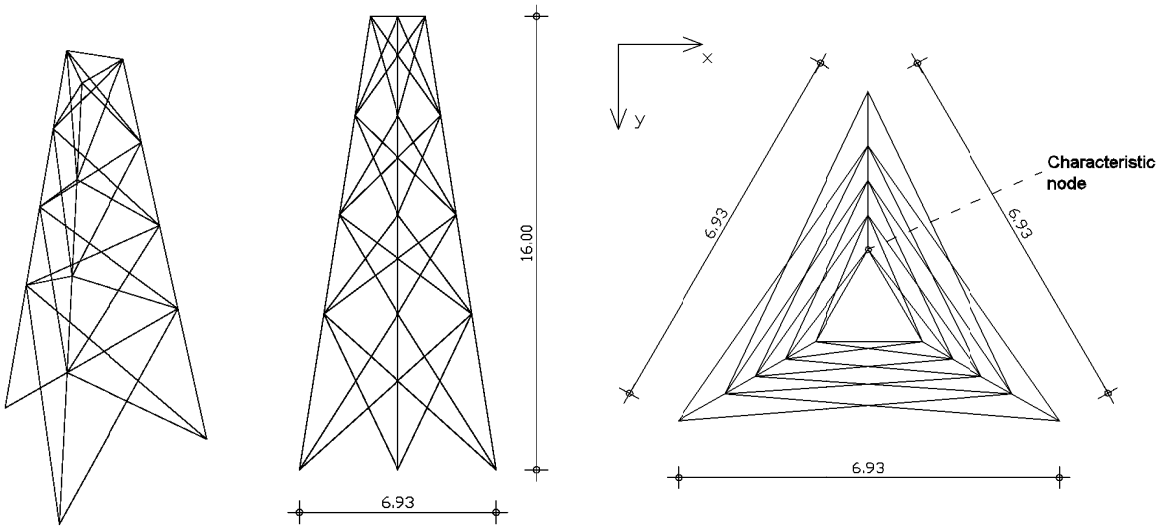


Figure 8.4 RDO - 39-bar space truss: (a) 3D view, (b) Side view, (c) Top view (dimensions in m).

Two objective functions are considered: the weight and the standard deviation of a characteristic nodal displacement representing the response of the structure, in this test example the variance of the top displacement (characteristic node) in *x* direction is selected.

The design variables considered are the dimensions of the members of the structure, four groups in total, taken from the *Circular Hollow Section* (CHS) table of the Eurocode (CEN 1993), shown in Table 8.3. For each design variable, two stochastic variables are assigned, the external diameter *d* and the thickness *t* of the circular hollow section. A vertical load V=2kN is applied to all nodes, while a probabilistic horizontal load F with a mean value 8kN is applied to the top nodes at the *x*-direction.

Table 8.3 RDO - 39-bar space truss: Circular Hollow Section of the Eurocode, Table 1 of 2.

Circular Hollow Section	Diameter (mm)	Thickness (mm)	Area (cm^2)	Moment of inertia (cm^4)	Circular Hollow Section	Diameter (mm)	Thickness (mm)	Area (cm^2)	Moment of inertia (cm^4)
21.3×2.3	21.3	2.3	1.373	0.6286	168.3×10	168.3	10	49.73	1564
26.9×2.3	26.9	2.3	1.778	1.356	168.3×12.5	168.3	12.5	61.18	1868
33.7×2.6	33.7	2.6	2.54	3.093	193.7×6.3	193.7	6.3	37.09	1630
33.7×2.9	33.7	2.9	2.806	3.357	193.7×8	193.7	8	46.67	2016
33.7×4	33.7	4	3.732	4.19	193.7×10	193.7	10	57.71	2442
42.4×2.6	42.4	2.6	3.251	6.464	193.7×12.5	193.7	12.5	71.16	2934
42.4×2.9	42.4	2.9	3.599	7.056	193.7×16	193.7	16	89.32	3554
42.4×4	42.4	4	4.825	8.991	219.1×7.1	219.1	7.1	47.29	2660
48.3×2.9	48.3	2.9	4.136	10.7	219.1×8	219.1	8	53.06	2960
48.3×3.2	48.3	3.2	4.534	11.59	219.1×10	219.1	10	65.69	3598
48.3×4	48.3	4	5.567	13.77	219.1×12.5	219.1	12.5	81.13	4345
60.3×2.9	60.3	2.9	5.229	21.59	219.1×16	219.1	16	102.1	5297
60.3×3.2	60.3	3.2	5.74	23.47	219.1×20	219.1	20	125.1	6261
60.3×4	60.3	4	7.075	28.17	244.5×7.1	244.5	7.1	52.95	3734
60.3×5	60.3	5	8.687	33.48	244.5×8	244.5	8	59.44	4160
76.1×2.9	76.1	2.9	6.669	44.74	244.5×10	244.5	10	73.67	5073
76.1×3.2	76.1	3.2	7.329	48.78	244.5×12.5	244.5	12.5	91.11	6147
76.1×4	76.1	4	9.06	59.06	244.5×16	244.5	16	114.9	7533
76.1×5	76.1	5	11.17	70.92	244.5×20	244.5	20	141.1	8957
88.9×3.2	88.9	3.2	8.616	79.21	273×7.1	273	7.1	59.31	5245
88.9×4	88.9	4	10.67	96.34	273×8	273	8	66.6	5852
88.9×5	88.9	5	13.18	116.4	273×10	273	10	82.62	7154
88.9×6.3	88.9	6.3	16.35	140.2	273×12.5	273	12.5	102.3	8697
88.9×8	88.9	8	20.33	168	273×16	273	16	129.2	10710
101.0×3.6	101.6	3.6	11.08	133.2	273×20	273	20	159	12800
101.0×5	101.6	5	15.17	177.5	323.9×7.1	323.9	7.1	70.66	8869
101.0×6.3	101.6	6.3	18.86	215.1	323.9×8.0	323.9	8	79.39	9910
101.6×8	101.6	8	23.52	259.5	323.9×10	323.9	10	98.61	12160
101.6×10	101.6	10	28.78	305.4	323.9×12.5	323.9	12.5	122.3	14850
114.3×3.6	114.3	3.6	12.52	192	323.9×16	323.9	16	154.8	18390
114.3×5	114.3	5	17.17	256.9	323.9×20	323.9	20	190.9	22140
114.3×6.3	114.3	6.3	21.38	312.7	355.6×8	355.6	8	87.36	13200
114.3×8	114.3	8	26.72	379.5	355.6×10	355.6	10	108.6	16220
114.3×10	114.3	10	32.77	449.7	355.6×12.5	355.6	12.5	134.7	19850
139.7×4	139.7	4	17.05	392.9	355.6×16	355.6	16	170.7	24660
139.7×5	139.7	5	21.16	480.5	355.6×20	355.6	20	210.9	29790

Table 8.4 RDO - 39-bar space truss: Circular Hollow Section of the Eurocode, Table 2 of 2.

Circular Hollow Section	Diameter (mm)	Thickness (mm)	Area (cm^2)	Moment of inertia (cm^4)	Circular Hollow Section	Diameter (mm)	Thickness (mm)	Area (cm^2)	Moment of inertia (cm^4)
139.7×6.3	139.7	6.3	26.4	588.6	406.4×8.8	406.4	8.8	109.9	21730
139.7×8	139.7	8	33.1	720.3	406.4×10	406.4	10	124.5	24480
139.7×10	139.7	10	40.75	861.9	406.4×12.5	406.4	12.5	154.7	30030
139.7×12.5	139.7	12.5	49.95	1020	406.4×16	406.4	16	196.2	37450
168.3×4.5	168.3	4.5	23.16	777.2	406.4×20	406.4	20	242.8	45430
168.3×6.3	168.3	6.3	32.06	1053	610×16	610	16	298.6	131800
168.3×8	168.3	8	40.29	1297	635×16	635	16	311.1	149100

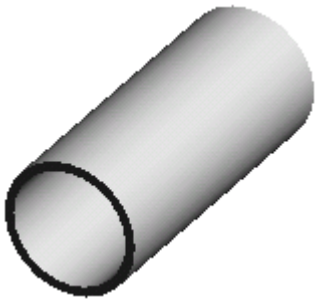


Figure 8.5 RDO - 39-bar space truss: 3D view of a Circular Hollow Section of the Eurocode.

Three types of constraints are imposed to the sizing optimization problem: (i) stress; (ii) compression force (for buckling); and (iii) displacement constraints, as imposed by the European design code (CEN 1993). The constraints are the same as the ones used for the numerical application of Section 8.1.1. Stress constraints are described in Eqs. (8.1) and (8.2), buckling constraints are described in Eqs. (8.3), (8.4) and (8.5), while the displacement constraint is described in Eq. (8.6). A constraint of 200mm on the maximum deflection is imposed. The type of *Probability Density Functiosn* (PDFs), the mean value, and the variance of the random parameters are shown in Table 8.5.

For this test case the $(\mu+\lambda)$-ES approach is used with $\mu=\lambda=5$, while a sample size of 1000 simulations is taken for the MCS. The multi-objective optimization problem is solved with the LWM, where all objectives are combined into a single scalar parameterized objective using weight coefficients, as described in Section 4.7.1. The resultant Pareto front curve is depicted in Figure 8.6, with the weight of the structure and the standard deviation of the horizontal displacement on the horizontal and vertical axis, respectively.

Table 8.5 RDO - 39-bar space truss: Characteristics of the random variables.

Symbol (units)	Description	PDF	Mean value μ	Standard Deviation σ	σ/μ	95% of values within
E (kN/m^2)	Young's Modulus	Normal	2.10×10^8	1.50×10^7	7.14%	$(1.81\times10^8, 2.39\times10^8)$
σ_y (kN/m^2)	Allowable stress	Normal	355000	35500	10.00%	$(2.85\times10^5, 4.25\times10^5)$
F (kN)	Horizontal loading	Normal	8	3	37.50%	(2.12, 13.88)
d	CHS Diameter	Normal	d_i *	$0.02\cdot d_i$	2%	$(0.9608\cdot d_i, 1.0392\cdot d_i)$
t	CHS Thickness	Normal	t_i *	$0.02\cdot t_i$	2%	$(0.9608\cdot t_i, 1.0392\cdot t_i)$

* Taken from the Circular Hollow Section table of the Eurocode, for every design.

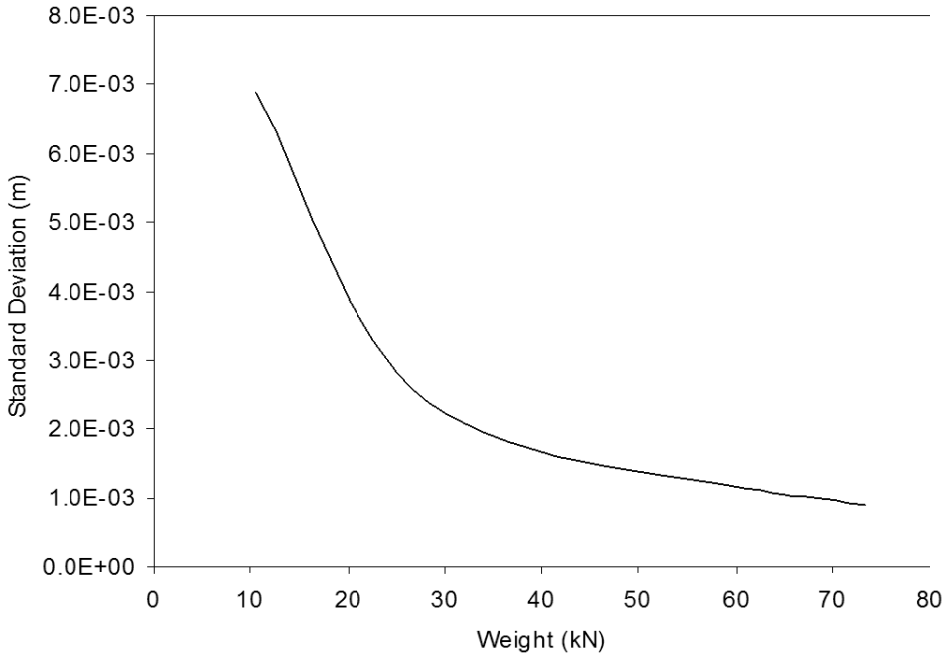


Figure 8.6 RDO - 39-bar space truss: Pareto front curve.

From the Pareto front curve the difference between DDO and RDO optimum designs is demonstrated in terms of the structural weight, the variance of the response and the probability of violation of the constraints. The upper left end of the Pareto Front corresponds to the deterministic optimum where the weight of the structure is the dominant criterion, while the lower right end of the curve corresponds to the optimum when the standard deviation of the response is considered as the dominant criterion. The intermediate Pareto optimal solutions are compromise solutions between these two extreme optimum designs under conflicting criteria.

8.1.3 Conclusions

In order to account for the response of the structure that is affected by the randomness of structural parameters, an RDO formulation of the optimization problem has to be used, as shown in this work. The Pareto front curves obtained in the two test examples show a strong conflict between the two objective functions in question. The deterministic formulation leads to an optimum design having the minimum weight, yet the maximum variance of the response. This wide variation of the response might cause the violation of some constraints that could consequently put the quality of the final design in doubt.

The proposed ES methodology combined with the standard LWM for treating multi-objective robust design optimization problems proved to be an efficient and reliable optimization tool, as it managed to generate good quality Pareto front curves.

8.2 Robust Design Optimization (RDO) with the non-dominant CEATm methodology

In this section two test examples are examined, a 3D transmission tower and a space truss bridge. For both problems, a multi-objective optimization formulation is used, as in the previous section, where the objective is to minimize the weight of the structure and the variance of the structural response.

A discrete ES scheme is used, where the design variables (the members of the structures) are taken from Eurocode tables. The multi-objective optimization problem is solved with the proposed non-dominant CEATm multi-objective optimization methodology using the Tchebycheff metric, described in detail in Section 4.10.1, while for treating the induced uncertainties the MCS method of Section 2.6 equipped with the *Latin Hypercube Sampling* (LHS) technique of Section 2.8 is used.

The numerical tests are performed in three stages:

i. In the first stage the statistical methods used for the stochastic analysis are verified. The number of LHS simulations required for the calculation of the mean value and the standard deviation of the characteristic displacement representing the structural response is compared with the corresponding number required by the basic MCS.

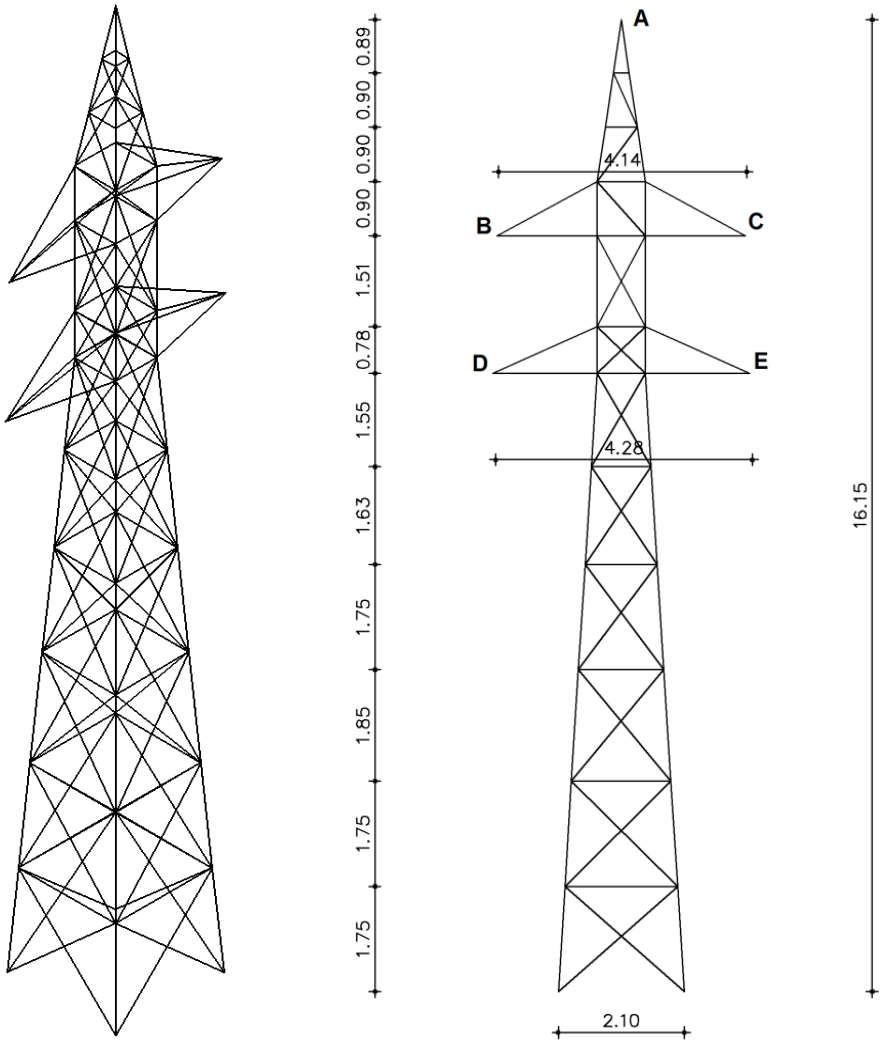
ii. In the second stage the advantages of the proposed non-dominant CEATm method over the LWM method are demonstrated through the comparison of the Pareto front curves obtained.

iii. At the third stage, the differences between DDO and RDO optimum designs, in terms of the final structural weight, the variance of response, the probability of violation of the constraints and the probability of failure, are illustrated.

The conclusions for both test examples are given together in Section 8.2.3.

8.2.1 3D Transmission tower – RDO test example

In this test example an RDO procedure is investigated in a 3D transmission tower example (Lagaros et al. 2005c; Plevris et al. 2005a; Plevris et al. 2005b). The transmission tower is depicted in Figures 8.7 and 8.8 together with its geometric characteristics. The nodal loads applied to the structure are shown in Table 8.6 in kN units.



(a) (b)

Figure 8.7 RDO - 3D Transmission tower: (a) 3D view, (b) Side view (dimensions in m).

Three types of constraints are imposed to the sizing optimization problem: (i) stress; (ii) compression force (for buckling); and (iii) displacement constraints. The constraints are the same as the ones used for the numerical application of Section 8.1.2. Stress constraints are described in Eqs. (8.1) and (8.2), buckling constraints are described in Eqs. (8.3), (8.4)and (8.5), while the displacement constraint is described in Eq. (8.6).

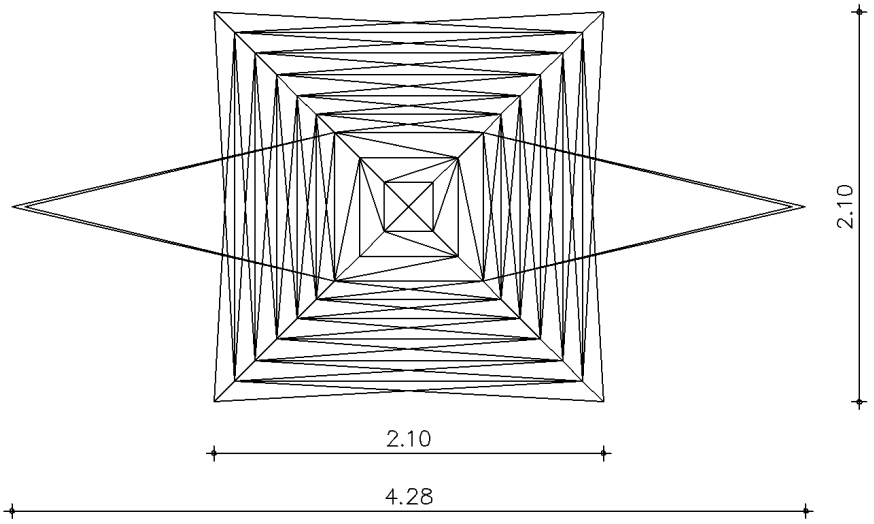


Figure 8.8 RDO - 3D Transmission tower: Top view (dimensions in m).

Table 8.6 RDO - 3D Transmission tower: Nodal loads (in kN units).

	F_x	F_y	F_z
Node *A*	-8.51	0.00	-4.82
Node *B*	-9.77	0.00	-5.36
Node *C*	-9.77	0.00	-5.36
Node *D*	-10.70	0.00	-5.36
Node *E*	-10.70	0.00	-5.36

The design variables considered are the dimensions of the structural members, divided into seven groups, taken from the Equal Angle Section table of the Eurocode (CEN 1993), shown in Table 8.1. For each design variable, two stochastic variables are assigned: the length L and the width t of the legs of the section.

The type of probability density function, the mean value, and the variance of the random parameters are given in Table 8.7. A maximum deflection of 200mm is imposed as a constraint for all nodes.

Table 8.7 RDO - 3D Transmission tower: Characteristics of the random variables.

Symbol (units)	Description	Probability Density Function	Mean value μ	Standard Deviation σ	σ/μ	95% of values within
E (kN/m^2)	Young's Modulus	Normal	2.10×10^8	1.50×10^7	7.14%	(1.81×10^8, 2.39×10^8)
σ_y (kN/m^2)	Allowable stress	Normal	355000	35500	10.00%	(2.85×10^5, 4.25×10^5)
F (kN)	Nodal loading	Normal	μ_F	$0.05\cdot\mu_F$	5%	($0.902\cdot\mu_F$, $1.098\cdot\mu_F$)
L	Legs length	Normal	L_i *	$0.02\cdot L_i$	2%	($0.961\cdot L_i$, $1.039\cdot L_i$)
t	Legs width	Normal	t_i *	$0.02\cdot t_i$	2%	($0.961\cdot t_i$, $1.039\cdot t_i$)

* Taken from the Equal Angle Section table of the Eurocode for every design.

Efficiency of the stochastic analysis method

In the first stage of the numerical study, the performance of the LHS procedure in calculating the statistical parameters required during the RDO procedure is compared to the basic MCS method. The influence of the number of simulations on the computed value of the variance of the top horizontal displacement is investigated.

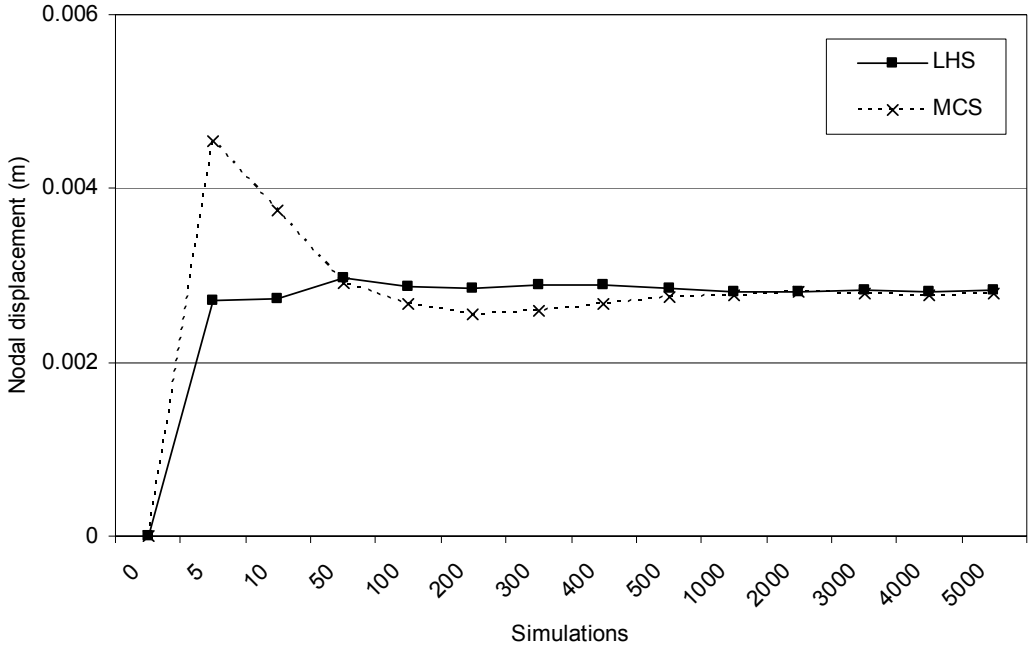


Figure 8.9 RDO - 3D Transmission tower: Efficiency of the LHS compared to the MCS in calculating the standard deviation of the structural response.

The results shown in Figure 8.9, for a randomly selected design, demonstrate the efficiency of the implemented LHS procedure. It can be seen that for the transmission

tower, 100 LHS compared to 500 MCS simulations are required in order to calculate the standard deviation of the structural response.

Comparison between LWM and CEATm

In the second stage of this study the advantages of the cascade evolutionary multi-objective optimization scheme using the Tchebycheff metric over the LWM are demonstrated. As was mentioned in Section 4.6, the quality of the Pareto front curve can be assessed by the number of Pareto optimum solutions obtained and their distribution along the front curve. Well distributed solutions along the curve give an indication of the efficiency of the multi-objective optimization method employed. The main drawback of the multi-objective optimization methods using scalarizing functions, such as the LWM, is that it is difficult to fulfil these two requirements.

For the comparative study performed, the robust design optimization problem considered has been solved with the LWM method and the proposed non-dominant CEATm multi-objective optimization scheme. For both test examples the LWM method has been implemented through two different runs with 10 and 30 points using the ES(μ+λ) optimization algorithm where μ=λ=5 are the number of parents and offspring, respectively. For the non-dominant CEATm$(\mu+\lambda)_{nrun,csteps}$ optimization scheme the corresponding parameters are μ=λ=5, *nrun*=10 and *csteps*=3. The resultant Pareto front curves are depicted in Figures 8.10 and 8.11 for LWM and Figure 8.12 for the CEATm. The horizontal axis corresponds to the structural weight and the vertical axis to the standard deviation of the characteristic node displacement. The RDO multi-objective optimization problem seems to be non-convex and the weakness of the LWM is obvious from the front curves of Figures 8.10 and 8.11. Well distributed pairs of weighting coefficients cannot guarantee equally well distributed Pareto optimum solutions along the front curve. On the other hand, the proposed CEATm optimization scheme manages to generate the Pareto front curve having a good distribution of the Pareto solutions along the front curve, as can be seen in Figure 8.12.

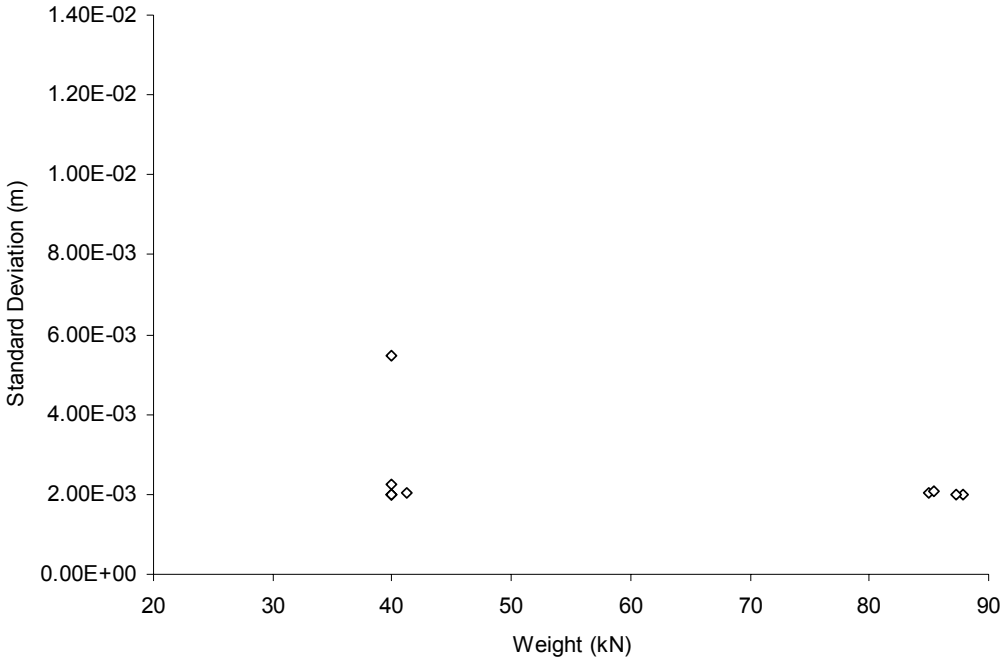


Figure 8.10 RDO - 3D Transmission tower: The Pareto front curve obtained with the LWM and 10 points.

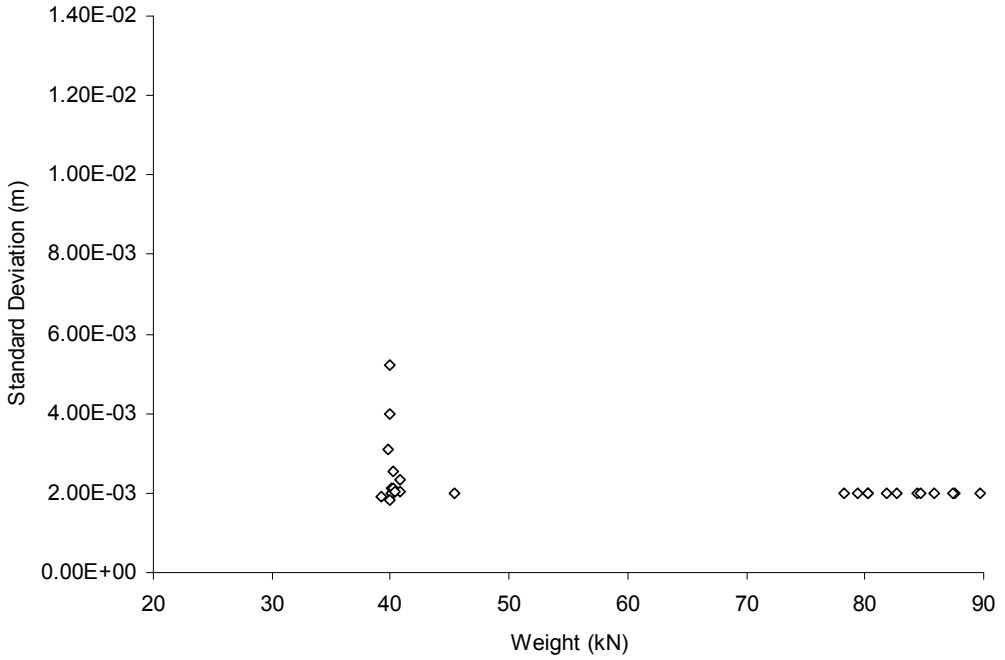


Figure 8.11 RDO - 3D Transmission tower: The Pareto front curve obtained with the LWM and 30 points.

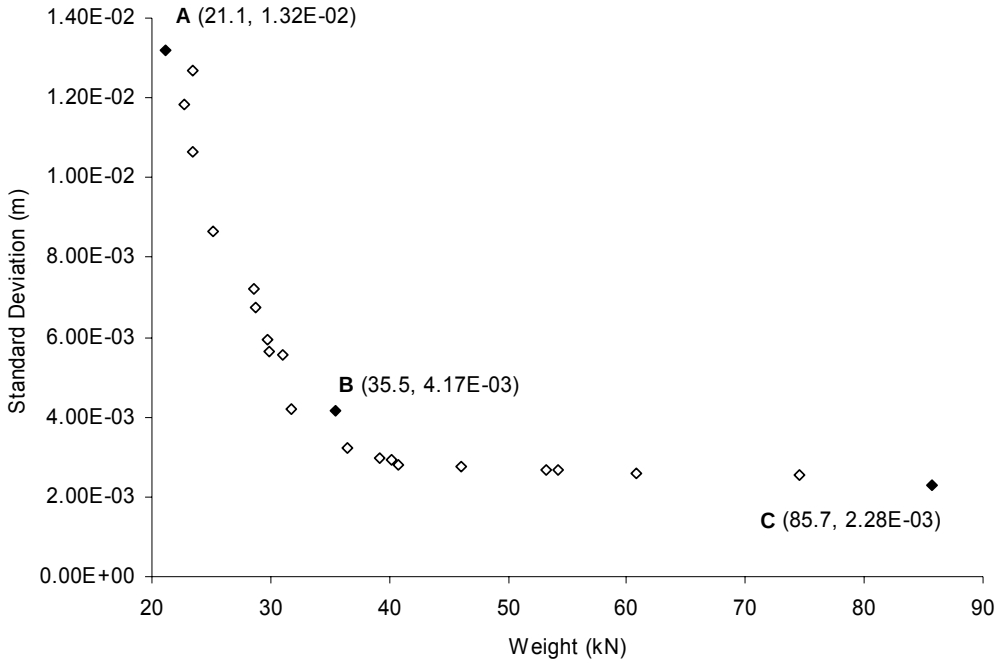


Figure 8.12 RDO - 3D Transmission tower: The Pareto front curve obtained with the non-dominant CEATm.

Comparison between DDO and RDO solutions

In the third stage of this study the difference between DDO and RDO optimum designs is demonstrated in terms of the structural weight, the variance of the response and the probability of violation of the constraints. The resultant Pareto front curve, when the proposed optimization scheme is used, is shown in Figure 8.12. The two ends of the Pareto front curve represent two extreme designs. Point *A* corresponds to the deterministic optimum where the weight of the structure is the dominant criterion. Point *C* is the optimum when the standard deviation of the response is considered as the dominant criterion. The intermediate Pareto optimal solutions are compromise solutions between these two extreme optimum designs under conflicting criteria.

In Table 8.8 a comparison is performed for the three optimum designs *A*, *B* and *C* of Figure 8.12. The RDO(*B*) optimum design is achieved considering a compromise between the weight and the standard deviation. An important outcome of this investigation is that the DDO optimum design violates the constraints with probability equal to 1.1% and probability of failure equal to 0.6%. On the other hand, the probability of violation and the probability of failure, in the case of the compromise optimum design B, are computed one to two orders of magnitude lower compared to those corresponding to DDO designs. As a consequence of this reduced probability of violation and failure, an increase of 70% on the optimum weight achieved is observed in the case of RDO compared to the DDO. The value of the probability of violation is significantly lower in the case of optimum design *C* where the corresponding probability is 0.002%. However, the optimum weight achieved is four times more than the one obtained with the DDO formulation.

Table 8.8 RDO - 3D Transmission tower: Characteristic optimal solutions.

Property	DDO (*A*)	RDO (*B*)	RDO (*C*)
Section 1 (*L*×*t*)*	80×8	110×10	150×12
Section 2 (*L*×*t*)*	70×7	150×14	160×15
Section 3 (*L*×*t*)*	80×6	100×8	180×16
Section 4 (*L*×*t*)*	70×9	80×6	150×12
Section 5 (*L*×*t*)*	70×6	80×8	150×12
Section 6 (*L*×*t*)*	75×7	90×7	150×12
Section 7 (*L*×*t*)*	75×8	100×8	160×17
Weight (kN)	21.1	35.5	85.7
Standard Deviation (m)	1.32×10^{-2}	4.17×10^{-3}	2.28×10^{-3}
p_{viol} (%)	1.1	7.0×10^{-2}	2.0×10^{-3}
p_f (%)	0.6	1.0×10^{-2}	0.8×10^{-3}

* Taken from the Equal Angle Section table of the Eurocode for every design.

Computational performance

The hardware platform that was used in this work consisted of a parallel computing implementation, a PC cluster with 25 nodes (Pentium III in 500 Mhz) interconnected through Fast Ethernet, with every node in a separate 100 Mbit/sec switch port. Message passing is performed with the programming platforms *Parallel Virtual Machine* (PVM) working over Fast Ethernet. Two parallel processing schemes have been considered:

i. *Parallel 1* corresponding to the exploitation of the parallel implementation of the optimization scheme;

ii. *Parallel 2* corresponding to the parallel implementation of the stochastic analysis involved in the optimization procedure.

The computational performance for obtaining the multi-objective RDO Pareto front curve is compared in Table 8.9. The solution of the single-objective DDO(*A*) and RDO(*C*) problems, in sequential and parallel computing environments is examined. It can be seen that the computational time required for obtaining the RDO(*C*) optimum solution is two orders of magnitude more than the corresponding time required to obtain the DDO(*A*) optimum solutions in sequential computing environment. This difference is reduced to one order of magnitude in the parallel computing environment.

Table 8.9 RDO - 3D Transmission tower: Computational performance.

Formulation	Optimization Scheme	Generations	FE analyses	Time (s)		
				Sequential	Parallel 1*	Parallel 2*
DDO (*A*)	CEA(5+5)	103	627	63	19	-
RDO (*C*)	CEATm(5+5)1,3	109	576	5214	349	229
RDO Pareto Front	CEATm(5+5)10,3	947	5528	51092	3127	2259

* In 25 processors.

8.2.2 Space truss bridge – RDO test example

The proposed RDO methodology is investigated in a pedestrian truss bridge example (Lagaros et al. 2005c). The truss bridge is depicted in Figures 8.13 and 8.14 together with its geometric characteristics. A 3D representation of the structure is given in Figure 8.15. Again, the numerical test is performed in three stages. In the first stage the statistical methods used for the stochastic analysis are verified. The number of LHS simulations required for the calculation of the mean value and the standard deviation of the characteristic displacement representing the structural response is compared with the corresponding number required by the basic MCS. In the second stage the advantages of the proposed non-dominant CEATm method over the LWM method are demonstrated through the comparison of the Pareto front curves obtained. At the third stage, the differences between DDO and RDO optimum designs, in terms of the final structural weight, the variance of response, the probability of violation of the constraints and the probability of failure, are illustrated.

The design variables considered are the dimensions of the structural members divided into twelve groups, taken from the double Equal Angle Section table of the Eurocode (CEN 1993). The single EAS sections are shown in Table 8.1.

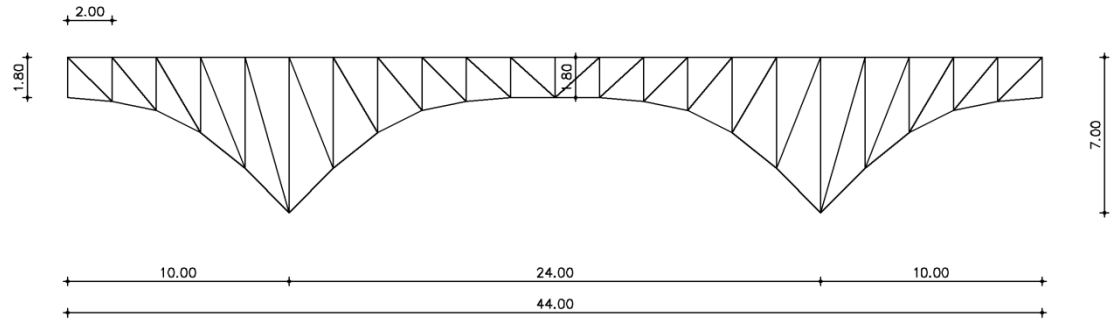


Figure 8.13 RDO - Space truss bridge: Side view.

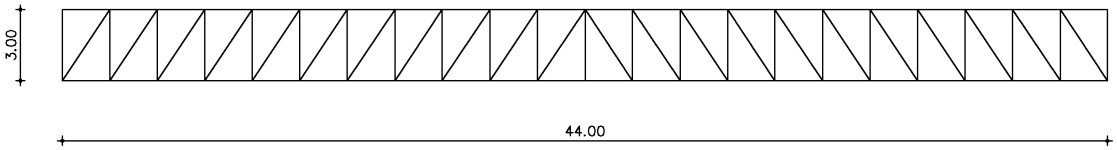


Figure 8.14 RDO - Space truss bridge: Top view.

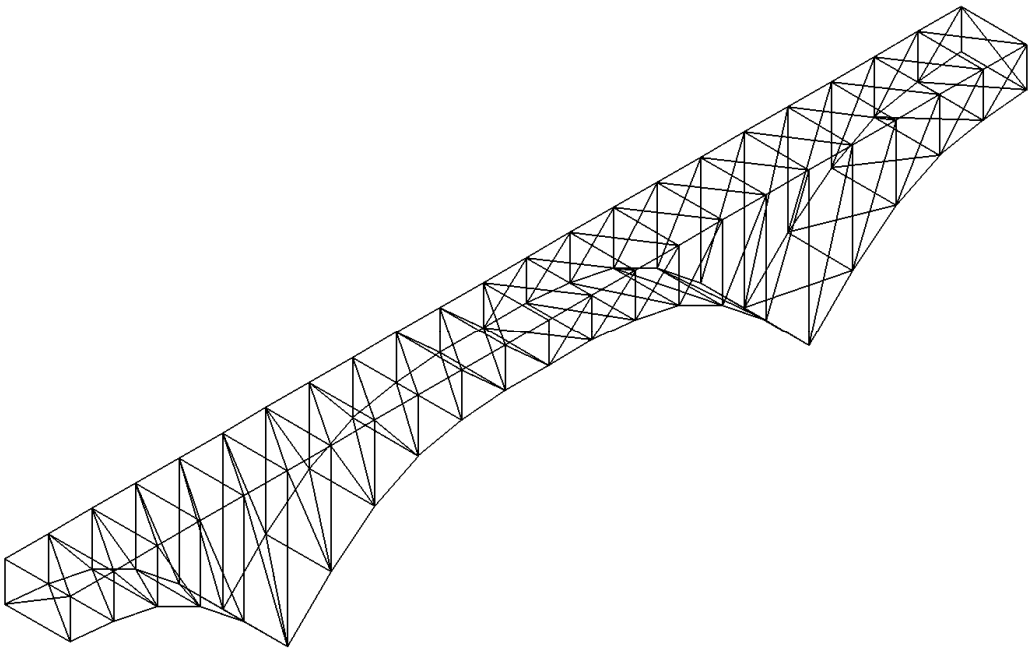


Figure 8.15 RDO - Space truss bridge: 3D view of the model.

For each design variable, three stochastic variables are assigned: the length L, the width t of the legs and the distance d between the two identical equal angle sections. The applied loading consists of:

i. Distributed load equal to 5kN/m² (dead load);

ii. Live loads (visiting vehicle);

iii. Wind actions according to Eurocode (CEN 1991; CEN 1993).

The type of PDFs, the mean value, and the variance of the random parameters are given in Table 8.10, while a maximum deflection constraint of 200mm is imposed on the nodes of the structure. Three types of constraints are imposed to the sizing optimization problem: (i) stress; (ii) compression force (for buckling); and (iii) displacement constraints. The constraints are the same as the ones used for the numerical application of Section 8.1.2. Stress constraints are described in Eqs. (8.1) and (8.2), buckling constraints are described in Eqs. (8.3), (8.4) and (8.5), while the displacement constraint is described in Eq. (8.6).

Table 8.10 RDO - Space truss bridge: Characteristics of the random variables.

Symbol (units)	Description	PDF	Mean value μ	Standard Deviation σ	σ/μ	95% of values within
E (kN/m^2)	Young's Modulus	Normal	2.10×10^8	1.50×10^7	7.14%	(1.81×10^8, 2.39×10^8)
σ_y (kN/m^2)	Allowable stress	Normal	355000	35500	10.00%	(2.85×10^5, 4.25×10^5)
F_P (kN)	Permanent loading	Normal	μ_{Fp}	$0.05\cdot\mu_{Fp}$	5%	($0.902\cdot\mu_{Fp}$, $1.098\cdot\mu_{Fp}$)
F_L (kN)	Live loading	Normal	μ_{FL}	$0.05\cdot\mu_{FL}$	5%	($0.902\cdot\mu_{FL}$, $1.098\cdot\mu_{FL}$)
F_W (kN)	Wind loading	Normal	μ_{Fw}	$0.10\cdot\mu_{Fw}$	10%	($0.804\cdot\mu_{Fw}$, $1.196\cdot\mu_{Fw}$)
L	Legs length	Normal	L_i *	$0.02\cdot L_i$	2%	($0.961\cdot L_i$, $1.039\cdot L_i$)
t	Legs width	Normal	t_i *	$0.02\cdot t_i$	2%	($0.961\cdot t_i$, $1.039\cdot t_i$)
d	EAS section Distance	Normal	d_i *	$0.02\cdot d_i$	2%	($0.961\cdot d_i$, $1.039\cdot d_i$)

* Taken from the double Equal Angle Section table of the Eurocode for every design.

Efficiency of the stochastic analysis method

In the first stage of the numerical study, the performance of the LHS procedure in calculating the statistical parameters required during the RDO procedure is compared to the basic MCS method. The influence of the number of simulations on the computed value of the variance of a characteristic displacement, namely the vertical deflection of the middle node for the Pedestrian Bridge, is examined. The results, for a randomly selected design, shown in Figure 8.16 demonstrate the efficiency of the implemented LHS procedure. It can be seen from Figure 8.16 that for the truss bridge example 100 LHS compared to 1000 MCS simulations are required.

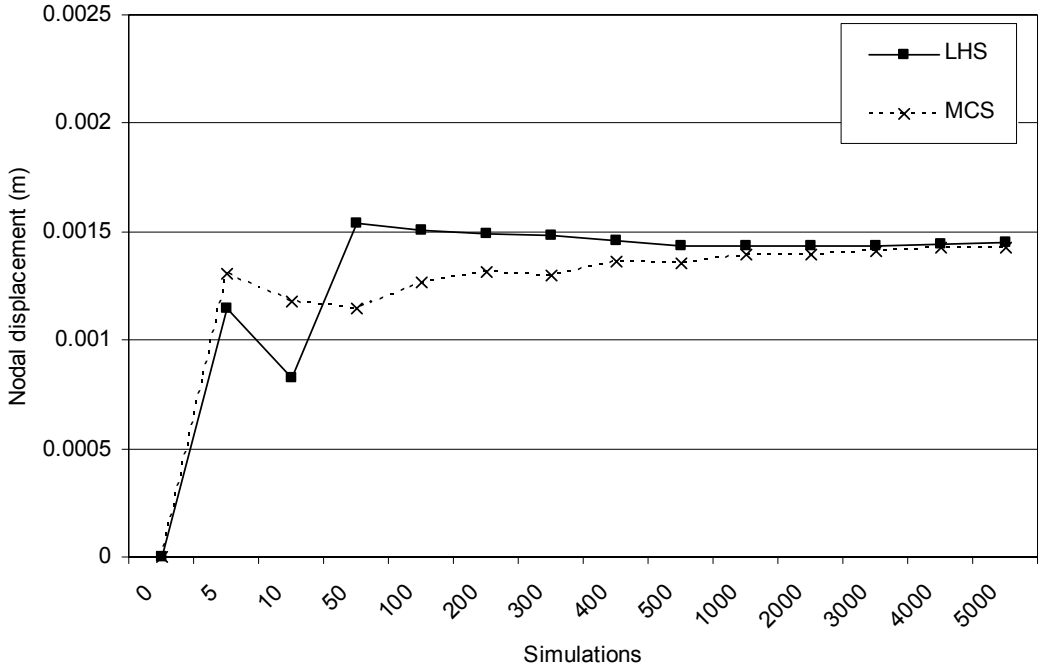


Figure 8.16 RDO - Space truss bridge: Efficiency of the LHS compared to the MCS in calculating the standard deviation of the structural response.

Comparison between LWM and CEATm

In the second stage of the study the robust design optimization problem has been solved with the LWM method and the proposed non-dominant *CEATm* multi-objective optimization scheme. For both test examples the LWM method has been implemented through two different runs with 10 and 30 points using the ES(μ+λ) optimization algorithm where μ=λ=5 are the number of parents and offspring, respectively. For the non-dominant CEATm(μ+λ)$_{nrun,csteps}$ optimization scheme the corresponding parameters are μ=λ=5, *nrun*=10 and *csteps*=3. The resultant Pareto front curves are depicted in Figures 8.17 and 8.18 for LWM and Figure 8.19 for the CEATm. The horizontal axis corresponds to the structural weight and the vertical axis to the standard deviation of the characteristic node displacement. As in the previous example, the LWM fails to generate a good Pareto front, while the proposed CEATm optimization scheme manages to generate the Pareto front curve having a good distribution of the Pareto solutions along the front curve, as can be seen in Figure 8.19.

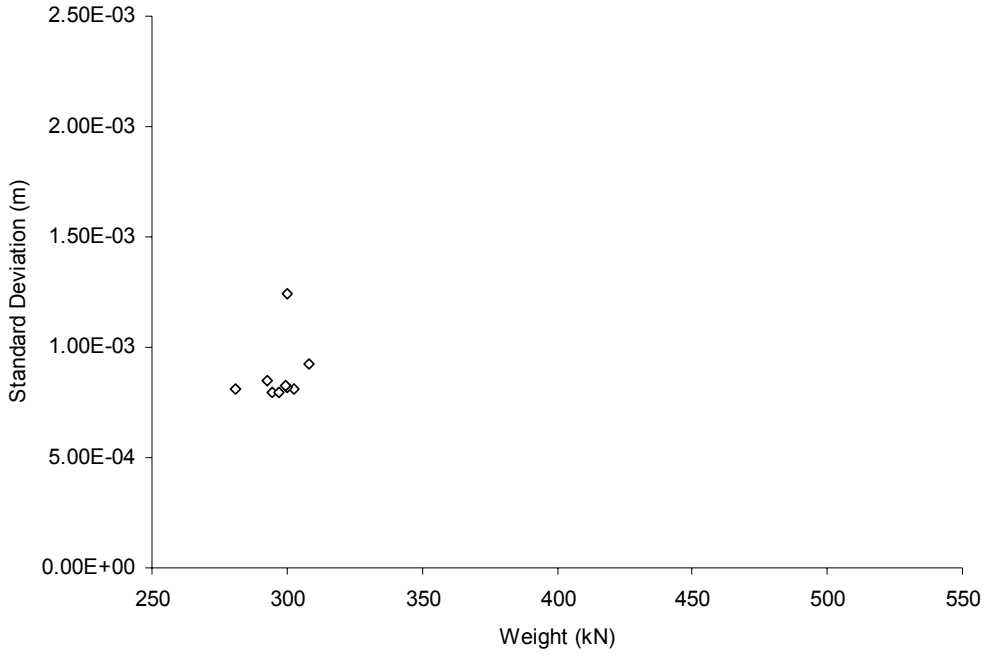


Figure 8.17 RDO - Space truss bridge:
The Pareto front curve obtained with LWM and 10 points.

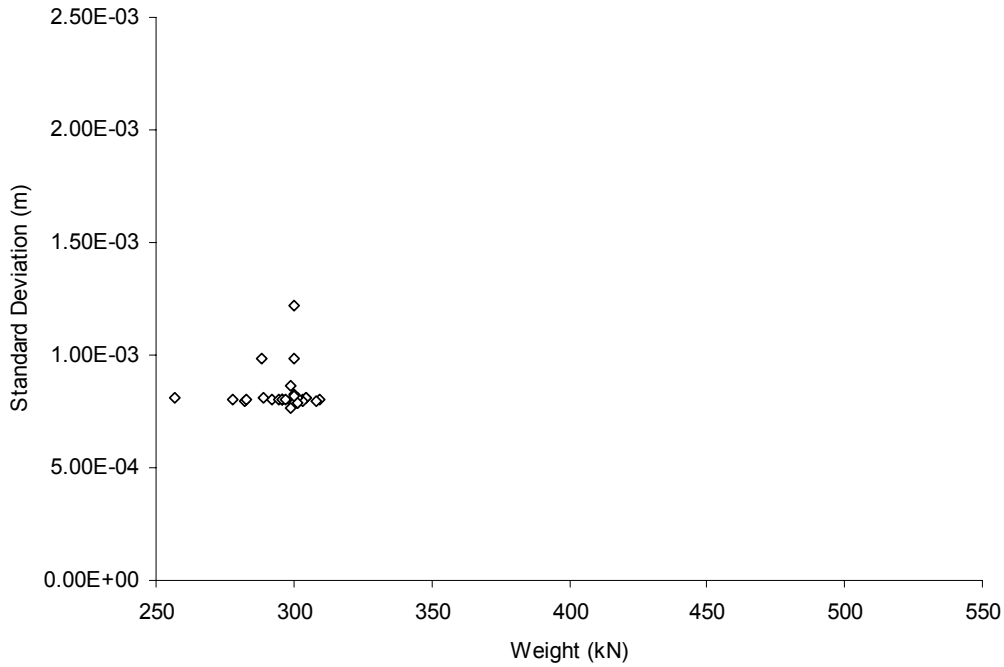


Figure 8.18 RDO - Space truss bridge:
The Pareto front curve obtained with LWM and 30 points.

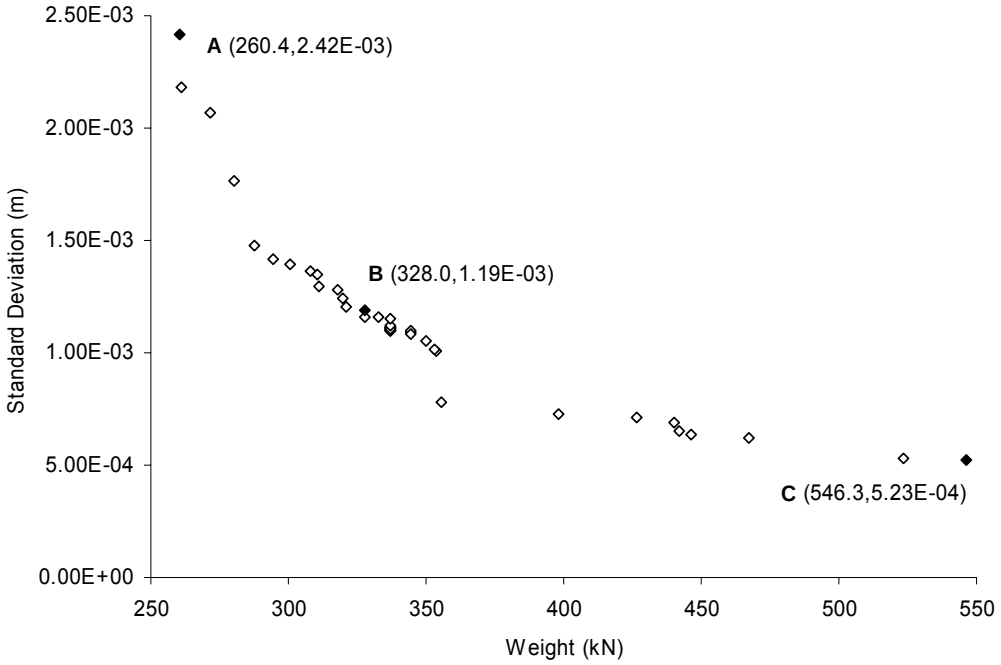


Figure 8.19 RDO - Space truss bridge:
The Pareto front curve obtained with the non-dominant CEATm.

Comparison between DDO and RDO solutions

In the third stage of this study the difference between DDO and RDO optimum designs is demonstrated in terms of the structural weight, the variance of the response and the probability of violation of the constraints. The resultant Pareto front curve when the proposed optimization scheme is used is shown in Figure 8.19. The two ends of the Pareto front curve represent two extreme designs. Point A corresponds to the deterministic optimum where the weight of the structure is the dominant criterion. Point C is the optimum when the standard deviation of the response is considered as the dominant criterion. The intermediate Pareto optimal solutions are compromise solutions between these two extreme optimum designs under conflicting criteria.

In Table 8.11 a comparison is performed for the three optimum designs *A*, *B* and *C* of Figure 8.19. The RDO(*B*) optimum design is achieved by considering a compromise between the weight and the standard deviation. The DDO optimum design violates the constraints with probability equal to 0.85% and probability of failure equal to 0.23%. On the other hand, the probability of violation and the probability of failure, in the case of the compromise optimum design B, are found to be one to two orders of magnitude lower compared to those corresponding to DDO designs. As a consequence of this reduced probability of violation and failure, an increase of 26% on the optimum weight achieved is observed in the case of RDO compared to the DDO. The value of the probability of violation is significantly lower in the case of optimum design C where the corresponding probability is 0.001%. However, the optimum weight achieved is two times heavier than the one obtained with the DDO formulation.

Table 8.11 RDO - Space truss bridge: Characteristic optimal solutions.

Property	DDO (*A*)	RDO (*B*)	RDO (*C*)
Section 1 (*L*×*t*,*d*)*	100×10,20	150×15,20	150×15,20
Section 2 (*L*×*t*,*d*)*	100×10,20	150×15,18	200×20,20
Section 3 (*L*×*t*,*d*)*	120×12,18	120×12,18	180×15,20
Section 4 (*L*×*t*,*d*)*	100×10,20	120×12,20	150×15,20
Section 5 (*L*×*t*,*d*)*	120×12,20	100×10,20	120×12,20
Section 6 (*L*×*t*,*d*)*	100×10,20	100×10,20	120×12,20
Section 7 (*L*×*t*,*d*)*	120×12,20	120×12,18	180×15,20
Section 8 (*L*×*t*,*d*)*	100×10,20	100×10,20	200×20,20
Section 9 (*L*×*t*,*d*)*	100×10,20	100×10,20	150×15,20
Section 10 (*L*×*t*,*d*)*	100×10,20	100×10,20	200×20,20
Section 11 (*L*×*t*,*d*)*	100×10,20	100×10,20	120×12,20
Section 12 (*L*×*t*,*d*)*	100×10,18	100×10,20	100×10,20
Weight (kN)	260.4	328.0	546.3
Standard Deviation (m)	2.42×10^{-3}	1.19×10^{-3}	5.23×10^{-4}
p_{viol} (%)	8.5×10^{-1}	1.3×10^{-2}	1.0×10^{-3}
p_f (%)	2.3×10^{-1}	0.7×10^{-2}	1.4×10^{-4}

* Taken from the double Equal Angle Section table of the Eurocode for every design.

Computational performance

The hardware platform used in this work for the parallel computing is the same as the one used for the previous test example. The computational performance for obtaining the multi-objective RDO Pareto front curve is compared in Table 8.12. Again, the solution of the single-objective DDO(*A*) and RDO(*C*) problems, in sequential and parallel computing environments is examined. It can be seen that the computational time required for obtaining the RDO(*C*) optimum solution is two orders of magnitude more than the corresponding time required to obtain the DDO(*A*) optimum solutions in sequential computing environment. This difference is reduced to one order of magnitude in the parallel computing environment.

Table 8.12 RDO - Space truss bridge: Computational performance.

Formulation	Optimization Scheme	Generations	FE analyses	Time (s)		
				Sequential	Parallel 1*	Parallel 2*
DDO (*A*)	CEA(5+5)	114	500	91	31	-
RDO (*C*)	CEATm(5+5)1,3	103	529	8643	552	368
RDO Pareto front curve	CEATm(5+5)10,3	863	4372	70673	4209	3055

* In 25 processors.

8.2.3 Conclusions

In order to account for the response of the structure that is affected by the randomness of structural parameters, a RDO formulation of the optimization problem has to be applied. It was shown that for the transmission tower test example, 100 LHS compared to 500 MCS simulations are required in order to calculate accurately the standard deviation of the structural response, while for the truss bridge test example, 100 LHS compared to 1000 MCS simulations are required. Thus, by applying the LHS scheme to the standard MCS method, the computational efficiency of the RDO methodology is improved significantly.

It was shown that for the specific test examples considered, the standard LWM using a scalarizing function did not manage to generate good quality Pareto Front curves in terms of the distribution of the solutions along the curve. On the other hand, the proposed non-dominant cascade evolutionary multi-objective optimization scheme using the Tchebycheff metric managed to generate well distributed solutions along the curve, showing the efficiency of the multi-objective optimization method employed.

From the Pareto front curves obtained, the difference between DDO and RDO optimum designs is demonstrated in terms of the structural weight, the variance of the response and the probability of violation of the constraints. The two ends of the resultant Pareto front curves represent two extreme designs. Point *A* of the Pareto Fronts of Figures 8.12 and 8.19 corresponds to the deterministic optimum where the weight of the structure is the dominant criterion. Point *C* is the optimum when the standard deviation of the response is considered as the dominant criterion. The intermediate Pareto optimal solutions (such as the one corresponding to point *B*) are compromise solutions between these two extreme optimum designs under conflicting criteria.

It was also shown that the computational effort required for obtaining a single RDO solution (such as solution *C*), is increased by two orders of magnitude more in sequential computing environment compared to that for obtaining a single-objective deterministic solution (such as solution *A*). In the parallel computing environment used in this study, this difference is reduced to one order of magnitude.

8.3 Reliability-Based Design Optimization (RBDO) assisted by Neural Networks

In this section, the Reliability-Based Design Optimization problem is examined in two test examples, a six-story plane frame and a six-story space frame. The probabilistic constraint is imposed on the probability of structural collapse due to successive formation of plastic hinges, taken equal to $p_a = 0.001$ for both test examples.

For the solution of the optimization problem, a $(\mu+\lambda)$-ES discrete scheme is used, with $\mu=\lambda=5$ for both test examples. The design variables are the cross sections of the structural members, taken from the Eurocode tables for the first test example and the American

tables for the second test example. The stochastic problem is solved with the MCS technique enhanced with the *Importance Sampling* (IS) methodology, described in detail in Section 2.9. A sample size of 500 is taken for the first test example, while for the second test example the sample size is 500, 1000 or 5000, in order to study the influence of the number of simulations on the quality of the optimization solution.

The three Reliability-Based Design Optimization methodologies assisted by Neural Networks have been used, where RBDO-NN*i* corresponds to the proposed RBDO with NN incorporating algorithm *i* (i=1,2,3), as described in detail in Sections 6.5.1, 6.5.2 and 6.5.3.

8.3.1 Six-story plane frame – RBDO test example with NN

The six-story plane frame depicted in Figure 8.20 (Papadrakakis et al. 2004a) has been considered in order to illustrate the efficiency of the proposed metamodel assisted methodology for reliability-based sizing optimization problems.

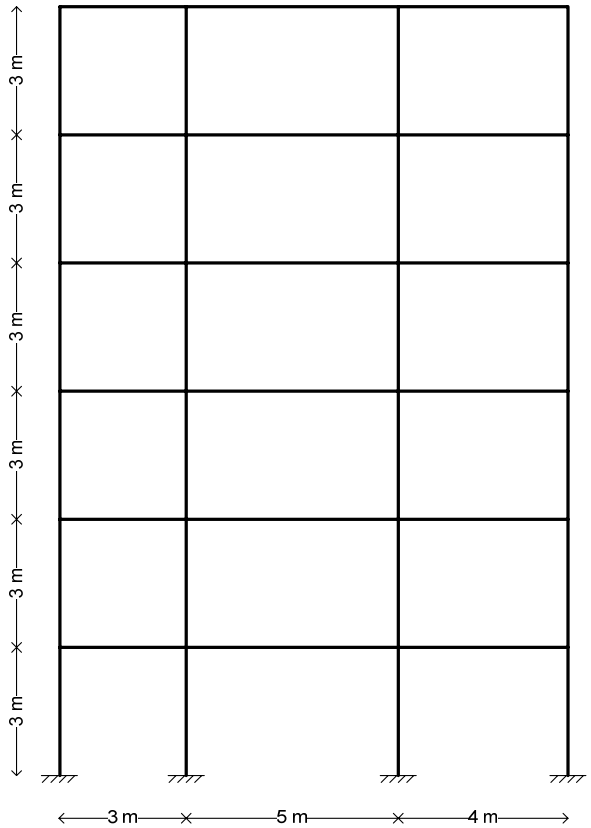


Figure 8.20 RBDO with NN - Six-story plane frame: View of the steel frame model.

The permanent load is taken as G=5kN/m² and the live load is taken as Q=2kN/m². The vertical loads are contributed from an effective area calculated with a depth of 5m perpendicular to the plane of the frame. The base shear is obtained from the EC8 (CEN 1998) response spectrum. Moreover, the importance factor γ_I has been taken equal to 1 and the characteristic periods T_A and T_B of the spectrum were considered equal to

0.15sec and 0.60sec, respectively. The damping correction factor is equal to 1.32, since a damping ratio of 2% has been considered. The dimensions and properties of the frame are depicted in Figure 8.20.

For beams, a capacity design against shear requires that the following condition is satisfied:

$$\frac{V_{G.Sd} + V_{M.Sd}}{V_{Pl.Rd}} \leq 0.5 \tag{8.7}$$

where $V_{G.Sd}$ is the shear force due to non seismic actions and $V_{M.Sd}$ is the shear force due to the application of resisting moments with opposite signs at the extremities of the beam. Moreover, the applied moment should be less than $M_{pl.Rd}$, while the axial load should be less than 15% of $N_{pl.Rd}$. For columns subject to bending with the presence of axial load the following formula should be satisfied:

$$\frac{N_{sd}}{\chi_{\min} \cdot N_{pl.Rd}} + \frac{\kappa_y \cdot M_{sd}}{M_{pl.Rd}} \leq 1 \tag{8.8}$$

where $\chi_{\min}$ is the reduction factor for flexural buckling taken equal to 0.7 and κ_y is a correction factor to allow for the combined effect of axial load and moment, taken equal to 1. Moreover the shear capacity should be double than the applied shear force. Plastic capacities for each member section are determined from the expression:

$$M_{pl,Rd} = \frac{W_{pl} \cdot f_y}{\gamma_{M0}} \tag{8.9}$$

$$N_{pl,Rd} = \frac{A \cdot f_y}{\gamma_{M1}} \tag{8.10}$$

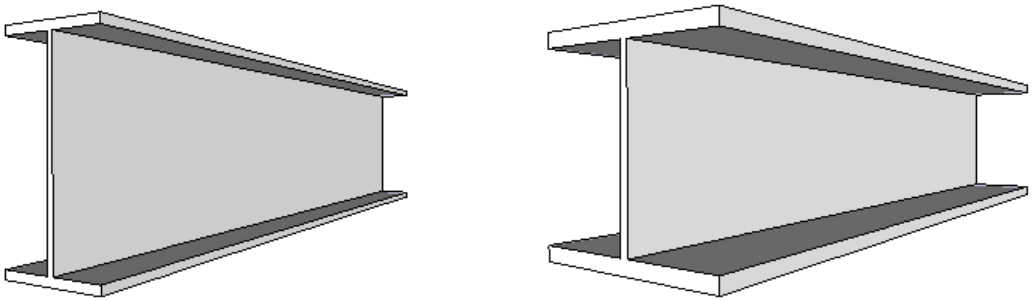
$$V_{pl,Rd} = \frac{1.04 \cdot h \cdot t_w \cdot f_y}{\sqrt{3} \cdot \gamma_{M0}} \tag{8.11}$$

where γ_{M0} and γ_{M1} are considered equal to 1.10 (CEN 1993). The inter-story drift constraint employed in a frame structure can be written as:

$$\frac{d_r}{\nu} \leq 0.006 \cdot h \tag{8.12}$$

where ν is a reduction factor for the serviceability limit state (taken equal to 2.5 for the test example considered in this study) and d_r is the relative drift between two consecutive stories. Different drift limits are adopted for the probabilistic and the deterministic constraints since failure is supposed to take place for considerably higher deformations. Furthermore, the strength ratio of column to beam is calculated and a check of whether the sections chosen are of class 1, as EC3 (CEN 1993) suggests, is also carried out. The section class check is necessary in order to ensure that the members of the structure have the capacity to develop their full plastic moment and rotational ductility, while the

strength ratio of column to beam is necessary in order to have a design consistent with the strong column-weak beam design philosophy of the seismic code.



(a) (b)

Figure 8.21 RBDO with NN - Six-story plane frame: 3D views of the European I-beams used: (a) IPE section for the beams, (b) HEB section for the columns.

The cross section of each member of the frame is assumed to be of European I-shape and one design variable is allocated for each member, while the objective function is the weight of the structure. The deterministic constraints are imposed on the inter-story drifts and for each group of structural members. The probabilistic constraint is imposed on the probability of structural collapse due to successive formation of plastic hinges and is set to p_a=0.001. The probability of failure caused by uncertainties related to material properties, geometry and loads of the structures is estimated using MCS with the Importance Sampling technique, described in detail in Section 2.9. The external loads, yield stress, modulus of elasticity and the dimensions of the cross-sections of the structural members are considered as random variables. The loads follow a log-normal probability density function, while random variables associated with material properties and cross-section dimensions follow a normal probability density function. The required importance sampling function $g_x(x)$ for the loads is assumed to follow a normal distribution. In Table 8.16 showing the results of the test examples, DDO stands for the conventional Deterministic Design Optimization approach, RBDO stands for the conventional *Reliability*-Based Design Optimization approach, while RBDO-NN*i* corresponds to the proposed Reliability-Based Design Optimization with NN incorporating algorithm *i* (i=1,2).

Table 8.13 RBDO with NN - Six-story plane frame: Characteristics of the random variables for the steel frame.

Random variable	Probability density function	Mean value	Standard deviation
E	Normal	200	$0.10 \cdot E$
σ_y	Normal	24.0	$0.10 \cdot \sigma_y$
Design variables	Normal	x_i	$0.1 \cdot x_i$
Loads ($G + 0.30 \cdot Q$)	Log- Normal	5.6	0.25

Table 8.14 RBDO with NN - Six-story plane frame: IPE sections of the Eurocode.

IPE Section	Height (mm)	Thickness (mm)	Thickness web (mm)	Thickness flange (mm)	Area (cm^2)	Moment of inertia I_y (cm^4)	Moment of inertia I_z (cm^4)
80	80	46	3.8	5.2	7.64	80.1	8.49
100	100	55	4.1	5.7	10.3	171	15.9
120	120	64	4.4	6.3	13.2	318	27.7
140	140	73	4.7	6.9	16.4	541	44.9
160	160	82	5	7.4	20.1	869	68.3
180	180	91	5.3	8	23.9	1317	101
200	200	100	5.6	8.5	28.5	1943	142
220	220	110	5.9	9.2	33.4	2772	205
240	240	120	6.2	9.8	39.1	3892	284
270	270	135	6.6	10.2	45.9	5790	420
300	300	150	7.1	10.7	53.8	8356	604
330	330	160	7.5	11.5	62.6	11770	788
360	360	170	8	12.7	72.7	16270	1043
400	400	180	8.6	13.5	84.5	23130	1318
450	450	190	9.4	14.6	98.8	33740	1676
500	500	200	10.2	16	116	48200	2142
550	550	210	11.1	17.2	134	67120	2668
600	600	220	12	19	156	92080	3387

The members of the structure are divided into two groups, each one having one design variable. The type of probability density functions, mean values, and variances of the random parameters are presented in Table 8.13. The mean value for each geometric variable (i.e. the cross-sectional dimensions) is taken as the value of the current design step of the corresponding variable x_i. The load-displacement curve of a node in the top floor of the frame is depicted in Figure 8.22, corresponding to the design vector (IPE300-HEB400). For this test example the $(\mu+\lambda)$-ES approach is used with $\mu=\lambda=5$, a sample size of 500 simulations has been examined for the MCS with the importance sampling technique (Papadrakakis and Lagaros 2002; Papadrakakis and Papadopoulos 1995).

Table 8.15 RBDO with NN - Six-story plane frame: HEB sections of the Eurocode.

HEB Section	Height (mm)	Thickness (mm)	Thickness web (mm)	Thickness flange (mm)	Area (cm^2)	Moment of inertia I_y (cm^4)	Moment of inertia I_z (cm^4)
100	100	100	6	10	26	450	167
120	120	120	6.5	11	34	864	318
140	140	140	7	12	43	1509	550
160	160	160	8	13	54.3	2492	889
180	180	180	8.5	14	65.3	3831	1363
200	200	200	9	15	78.1	5696	2003
220	220	220	9.5	16	91	8091	2843
240	240	240	10	17	106	11260	3923
260	260	260	10	17.5	118.4	14920	5135
280	280	280	10.5	18	131.4	19270	6595
300	300	300	11	19	149.1	25170	8563
320	320	300	11.5	20.5	161.3	30820	9239
340	340	300	12	21.5	170.9	36660	9690
360	360	300	12.5	22.5	180.6	43190	10140
400	400	300	13.5	24	197.8	57680	10820
450	450	300	14	26	218	79890	11720
500	500	300	14.5	28	238.6	107180	12620
550	550	300	15	29	254.1	136690	13080
600	600	300	15.5	30	270	171040	13530
650	650	300	16	31	286	210600	13980
700	700	300	17	32	306	256900	14440
800	800	300	17.5	33	334	369100	14900
900	900	300	18.5	35	371	494100	15820
1000	1000	300	19	36	400	644700	16280

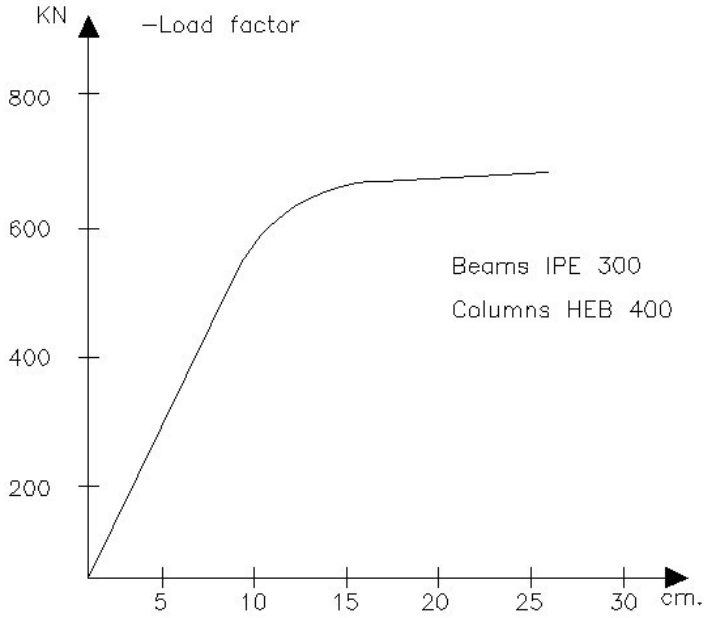


Figure 8.22 RBDO with NN - Six-story plane frame:
Load-displacement curve for the steel frame for a specific design.

Table 8.16 RBDO with NN - Six-story plane frame:
Performance of the methods for the steel frame.

Optimization procedure	Optimum Design (beam-column)	p_f**	Optimum weight (kN)	Time (h)
DDO	(IPE330 - HEB280)	0.88×10^{-1}	107	-
RBDO (500 simulations)	(IPE330 - HEB360)	0.113×10^{-2}	130	4.7
RBDO-NN1 (500 simulations.)	(IPE330 - HEB360)	0.113×10^{-2}	130	4.3
RBDO-NN2*	(IPE300 - HEB400)	0.905×10^{-3}	139	1.1
RBDO-NN3*	(IPE300 - HEB400)	0.905×10^{-3}	139	0.8

*For 100000 simulations.
**For 100000 simulations using the NN2 scheme.

As can be observed from Table 8.16 the computed probability of failure p_f for the deterministic optimum design is unacceptable since it exceeds substantially the accepted value 10^{-3}. On the other hand, the optimum weight achieved by the RBDO with 500 simulations is heavier by 21% compared to the deterministic one. For the application of the RBDO-NN1 methodology the number of NN input units is equal to the number of design variables, whereas one output unit is needed, according to both deterministic and probabilistic constraints. The output unit takes the values 1 or 0, corresponding to a feasible or infeasible design vector, respectively. Consequently the NN configuration implemented in this case has one hidden layer with 10 nodes resulting in a 2-10-1 NN architecture used for all runs. The training set consists of 100 training patterns chosen with the

requirement that the full range of the design space should be represented in the training procedure.

For the application of the RBDO-NN2 methodology the number of NN input units is equal to the number of the random variables, whereas one output unit is needed corresponding to the critical load factor. Consequently, the NN configuration results in a 3-7-1 NN architecture which is used for all runs. The number of step-by-step limit finite element analysis calculations performed for the training of NN is taken 60 corresponding to different groups of random variables properly selected from the random field. As can be seen in Table 8.16 the proposed RBDO-NN2 optimization scheme, with 100000 simulations, leads to a heavier design by 30% and 6.9% compared to the DDO and the RBDO with 500 simulations, respectively. As it can be seen, the number of MCS in the case of NN2 scheme can be extremely large without affecting its computational efficiency due to the trivial computing time required by the NN to perform one Monte Carlo simulation. In the case of the RBDO-NN3 methodology the advantages of the NN1 and NN2 methodologies are combined leading to the optimum design achieved by the RBDO-NN2 methodology but with less computing effort.

8.3.2 Six-story space frame – RBDO test example with NN

The test example involves the Reliability-Based Design Optimization of a three dimensional frame (Papadrakakis et al. 2004b; Papadrakakis et al. 2005; Plevris et al. 2006a). It is used in order to illustrate the efficiency of the proposed methodologies for treating realistic large-scale RBDO problems.

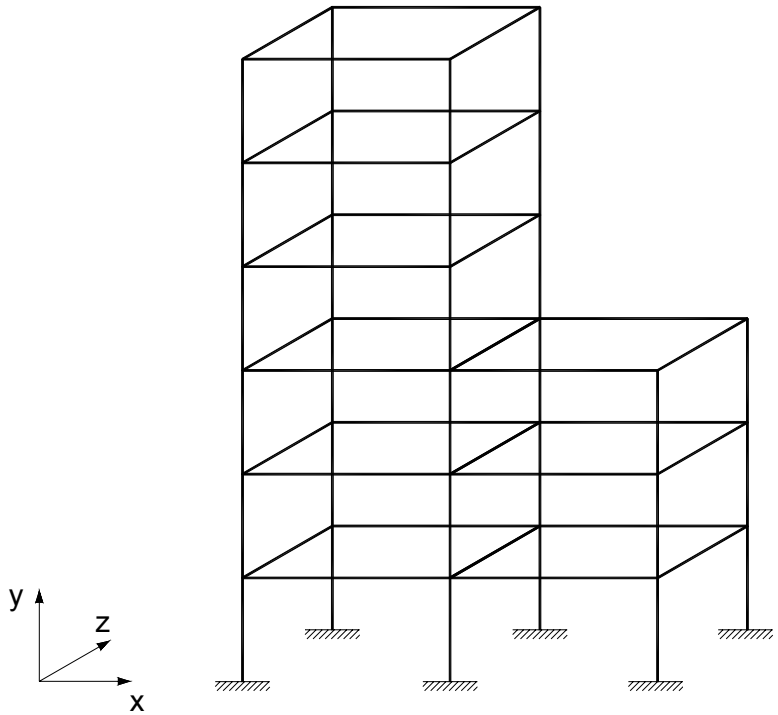


Figure 8.23 RBDO with NN - Six-story space frame: 3D view, side view and top view.

The cross section of each member is an American standard steel wide flange beam (W-shape) and one design variable is allocated for each member. The objective function is the weight of the structure to be minimized. The deterministic constraints are applied on the inter-story drifts and on the member stresses. This example consists of 63 elements with 180 degrees of freedom as shown in Figure 8.23

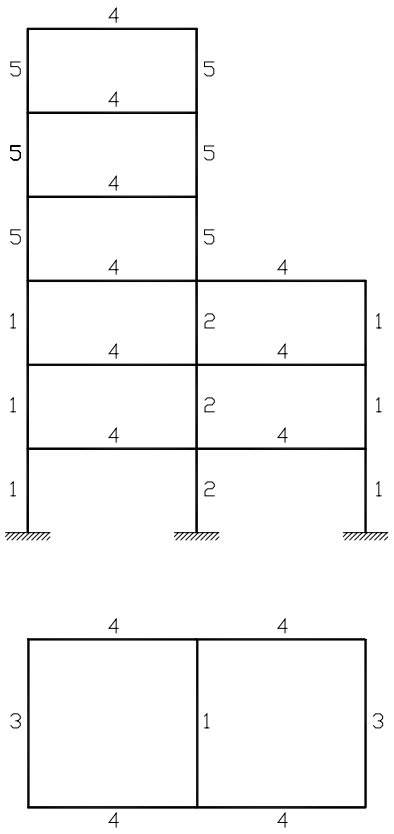


Figure 8.24 RBDO with NN - Six-story space frame:
(a) 3D model, (b) Element groups.

The structure is loaded with a gravity load of 19.16 kPa on all floor levels and a lateral load of 110 kN applied at each node in the front elevation along the z direction. The members of the structure are divided into five groups, as shown in Figure 8.23, each one assigned to one design variable. According to Eurocode 3 (CEN 1993) for the design of steel structures, the inter-story drift constraint employed for a frame structure can be written as:

$$\frac{d_r}{\nu} \leq 0.006 \cdot h \tag{8.13}$$

where ν is a reduction factor for the *Serviceability Limit State* (SLS), taken equal to 2.5 for this test example, and d_r is the relative drift between two consecutive stories. The stress constraint function for beams subjected to biaxial bending under compression, for each group of structural members, is given by the formula:

$$\frac{N_{sd}}{A\, f_y/\gamma_{M1}} + \frac{M_{sd,y}}{W_{pl,y}\, f_y/\gamma_{M1}} + \frac{M_{sd,z}}{W_{pl,z}\, f_y/\gamma_{M1}} \leq 1.0 \qquad (8.14)$$

where N_{sd}, $M_{sd,y}$, $M_{sd,z}$ are the computed stress resultants, $W_{pl,y}$, $W_{pl,z}$ are the plastic first moments of inertia, f_y is the yield stress and γ_{M1} is a safety factor, taken equal to 1.10 (CEN 1993).

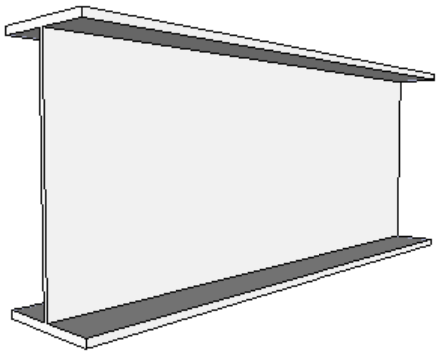


Figure 8.25 RBDO with NN - Six-story space frame:
3D view of the American standard steel wide flange beam (W-shape).

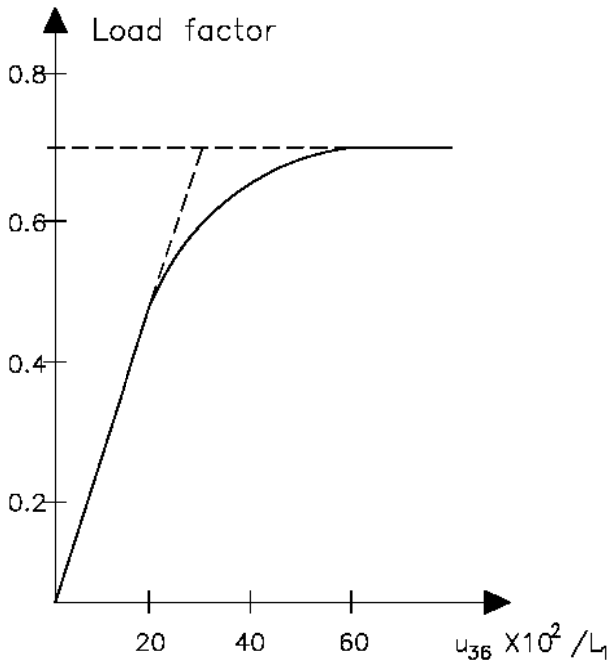


Figure 8.26 RBDO with NN - Six-story space frame: Load-displacement curve up to failure.

Table 8.17 RBDO with NN - Six-story space frame: American standard steel wide flange beam (W-shape) sections, Table 1 of 5.

Designation	Height (in)	Width (in)	Thickness web (in)	Thickness flange (in)	Area (in^2)	Moment of inertia I_y (in^4)	Moment of inertia I_z (in^4)
W 27 x 178	27.81	14.085	0.725	1.19	52.3	6990	555
W 27 x 161	27.59	14.02	0.66	1.08	47.4	6280	497
W 27 x 146	27.38	13.965	0.605	0.975	42.9	5630	443
W 27 x 114	27.29	10.07	0.57	0.93	33.5	4090	159
W 27 x 102	27.09	10.015	0.515	0.83	30	3620	139
W 27 x 94	26.92	9.99	0.49	0.745	27.7	3270	124
W 27 x 84	26.71	9.96	0.46	0.64	24.8	2850	106
W 24 x 162	25	12.955	0.705	1.22	47.7	5170	443
W 24 x 146	24.74	12.9	0.65	1.09	43	4580	391
W 24 x 131	24.48	12.855	0.605	0.96	38.5	4020	340
W 24 x 117	24.26	12.8	0.55	0.85	34.4	3540	297
W 24 x 104	24.06	12.75	0.5	0.75	30.6	3100	259
W 24 x 94	24.31	9.065	0.515	0.875	27.7	2700	109
W 24 x 84	24.1	9.02	0.47	0.77	24.7	2370	94.4
W 24 x 76	23.92	8.99	0.44	0.68	22.4	2100	82.5
W 24 x 68	23.73	8.965	0.415	0.585	20.1	1830	70.4
W 24 x 62	23.74	7.04	0.43	0.59	18.2	1550	34.5
W 24 x 55	23.57	7.005	0.395	0.505	16.2	1350	29.1
W 21 x 147	22.06	12.51	0.72	1.15	43.2	3630	376
W 21 x 132	21.83	14.44	0.65	1.035	38.8	3220	333
W 21 x 122	21.68	12.39	0.6	0.96	35.9	2960	305
W 21 x 111	21.51	12.34	0.55	0.875	32.7	2670	274
W 21 x 101	21.36	12.29	0.5	0.8	29.8	2420	248
W 21 x 93	21.62	8.42	0.58	0.93	27.3	2070	92.9
W 21 x 83	21.43	8.355	0.515	0.835	24.3	1830	81.4
W 21 x 73	21.24	8.295	0.455	0.74	21.5	1600	70.6
W 21 x 68	21.13	8.27	0.43	0.685	20	1480	64.7
W 21 x 62	20.99	8.24	0.4	0.615	18.3	1330	57.5
W 21 x 57	21.06	6.555	0.405	0.65	16.7	1170	30.6
W 21 x 50	20.83	6.53	0.38	0.535	14.7	984	24.4
W 21 x 44	20.66	6.5	0.35	0.45	13	843	20.7
W 18 x 119	18.97	11.265	0.655	1.06	35.1	2190	253
W 18 x 106	18.73	11.2	0.59	0.94	31.1	1910	220
W 18 x 97	18.59	11.145	0.535	0.87	28.5	1750	201
W 18 x 86	18.39	11.09	0.48	0.77	25.3	1530	175
W 18 x 76	18.21	11.035	0.425	0.68	22.3	1330	152
W 18 x 71	18.47	7.635	0.495	0.81	20.8	1170	60.3
W 18 x 65	18.35	7.59	0.45	0.75	19.1	1070	54.8
W 18 x 60	18.24	7.555	0.415	0.695	17.6	984	50.1

Table 8.18 RBDO with NN - Six-story space frame: American standard steel wide flange beam (W-shape) sections, Table 2 of 5.

Designation	Height (in)	Width (in)	Thickness web (in)	Thickness flange (in)	Area (in^2)	Moment of inertia I_y (in^4)	Moment of inertia I_z (in^4)
W 18 x 55	18.11	7.53	0.39	0.63	16.2	890	44.9
W 18 x 50	17.99	7.495	0.355	0.57	14.7	800	40.1
W 18 x 46	18.09	6.06	0.36	0.605	13.5	712	22.5
W 18 x 40	17.9	6.015	0.315	0.525	11.8	612	19.1
W 18 x 35	17.7	6	0.3	0.425	10.3	510	15.3
W 16 x 100	16.97	10.425	0.585	0.985	29.4	1490	186
W 16 x 89	16.75	10.365	0.525	0.875	26.2	1300	163
W 16 x 77	16.52	10.295	0.455	0.76	22.6	1110	138
W 16 x 67	16.33	10.235	0.395	0.665	19.7	954	119
W 16 x 57	16.43	7.12	0.43	0.715	16.8	758	43.1
W 16 x 50	16.26	7.07	0.38	0.63	14.7	659	37.2
W 16 x 45	16.13	7.035	0.345	0.565	13.3	586	32.8
W 16 x 40	16.01	6.995	0.305	0.505	11.8	518	28.9
W 16 x 36	15.86	6.985	0.295	0.43	10.6	448	24.5
W 16 x 31	15.88	5.525	0.275	0.44	9.12	375	12.4
W 16 x 26	15.69	5.5	0.25	0.345	7.68	301	9.59
W 14 x 730	22.42	17.89	3.07	4.91	215	14300	4720
W 14 x 665	21.64	17.65	2.83	4.52	196	12400	4170
W 14 x 605	20.92	17.415	2.595	4.16	178	10800	3680
W 14 x 550	20.24	17.2	2.38	3.82	162	9430	3250
W 14 x 500	19.6	17.01	2.19	3.5	147	8210	2880
W 14 x 455	19.02	16.835	2.015	3.21	134	7190	2560
W 14 x 426	18.67	16.695	1.875	3.035	125	6600	2360
W 14 x 398	18.29	16.59	1.77	2.845	117	6000	2170
W 14 x 370	17.92	16.475	1.655	2.66	109	5440	1990
W 14 x 342	17.57	16.36	1.54	2.47	101	4900	1810
W 14 x 311	17.12	16.23	1.41	2.26	91.4	4330	1610
W 14 x 283	16.74	16.11	1.29	2.07	83.3	3840	1440
W 14 x 257	16.38	15.995	1.175	1.89	75.6	3400	1290
W 14 x 233	16.04	15.89	1.07	1.72	68.5	3010	1150
W 14 x 211	15.72	15.8	0.98	1.56	62	2660	1030
W 14 x 193	15.48	15.71	0.89	1.44	56.8	2400	931
W 14 x 176	15.22	15.65	0.83	1.31	51.8	2140	838
W 14 x 159	14.98	15.565	0.745	1.19	46.7	1900	748
W 14 x 145	14.78	15.5	0.68	1.09	42.7	1710	677
W 14 x 132	14.66	14.725	0.645	1.03	38.8	1530	548
W 14 x 120	14.48	14.67	0.59	0.94	35.3	1380	495
W 14 x 109	14.32	14.605	0.525	0.86	32	1240	447

Table 8.19 RBDO with NN - Six-story space frame: American standard steel wide flange beam (W-shape) sections, Table 3 of 5.

Designation	Height (in)	Width (in)	Thickness web (in)	Thickness flange (in)	Area (in^2)	Moment of inertia I_y (in^4)	Moment of inertia I_z (in^4)
W 14 x 99	14.16	14.565	0.485	0.78	29.1	1110	402
W 14 x 90	14.02	14.52	0.44	0.71	26.5	999	362
W 14 x 82	14.31	10.13	0.51	0.855	24.1	882	148
W 14 x 74	14.17	10.07	0.45	0.785	21.8	796	134
W 14 x 68	14.04	10.035	0.415	0.72	20	723	121
W 14 x 61	13.89	9.995	0.375	0.645	17.9	640	107
W 14 x 53	13.92	8.06	0.37	0.66	15.6	541	57.7
W 14 x 48	13.79	8.03	0.34	0.595	14.1	485	51.4
W 14 x 43	13.66	7.995	0.305	0.53	12.6	428	45.2
W 14 x 38	14.1	6.77	0.31	0.515	11.2	385	26.7
W 14 x 34	13.98	6.745	0.285	0.455	10	340	26.3
W 14 x 30	13.84	6.73	0.27	0.385	8.85	291	19.6
W 14 x 26	13.91	5.025	0.255	0.42	7.69	245	8.91
W 14 x 22	13.74	5	0.23	0.335	6.49	199	7
W 12 x 336	16.82	13.385	1.775	2.955	98.8	4060	1190
W 12 x 305	16.32	13.235	1.625	2.705	89.6	3550	1050
W 12 x 279	15.85	13.14	1.53	2.47	81.9	3110	937
W 12 x 252	15.41	13.005	1.395	2.25	74.1	2720	828
W 12 x 230	15.05	12.895	1.285	2.07	67.7	2420	742
W 12 x 210	14.71	12.79	1.18	1.9	61.8	2140	664
W 12 x 190	14.38	12.67	1.06	1.735	55.8	1890	589
W 12 x 170	14.03	12.57	0.96	1.56	50	1650	517
W 12 x 152	13.71	12.48	0.87	1.4	44.7	1430	454
W 12 x 136	13.41	12.4	0.79	1.25	39.9	1240	398
W 12 x 120	13.12	12.32	0.71	1.105	35.3	1070	345
W 12 x 106	12.89	12.22	0.61	0.99	31.2	933	301
W 12 x 96	12.71	12.16	0.55	0.9	28.2	833	270
W 12 x 87	12.53	12.125	0.515	0.81	25.6	740	241
W 12 x 79	12.38	12.08	0.47	0.735	23.2	662	216
W 12 x 72	12.25	12.04	0.43	0.67	21.1	597	195
W 12 x 65	12.12	12	0.39	0.605	19.1	533	174
W 12 x 58	12.19	10.01	0.36	0.64	17	475	107
W 12 x 53	12.06	9.995	0.345	0.575	15.6	425	95.8
W 12 x 50	12.19	8.08	0.37	0.64	14.7	394	56.3
W 12 x 45	12.06	8.045	0.335	0.575	13.2	350	50
W 12 x 40	11.94	8.005	0.295	0.515	11.8	310	44.1
W 12 x 35	12.5	6.56	0.3	0.52	10.3	285	24.5
W 12 x 30	12.34	6.52	0.26	0.44	8.79	238	20.3

Table 8.20 RBDO with NN - Six-story space frame: American standard steel wide flange beam (W-shape) sections, Table 4 of 5.

Designation	Height (in)	Width (in)	Thickness web (in)	Thickness flange (in)	Area (in^2)	Moment of inertia I_y (in^4)	Moment of inertia I_z (in^4)
W 12 x 26	12.22	6.49	0.23	0.38	7.65	204	17.3
W 12 x 22	12.31	4.03	0.26	0.425	6.48	156	4.66
W 12 x 19	12.16	4.005	0.235	0.35	5.57	130	3.76
W 12 x 16	11.99	3.99	0.22	0.265	4.71	103	2.82
W 12 x 14	11.91	3.97	0.2	0.225	4.16	88.6	2.36
W 10 x 112	11.36	10.415	0.755	1.25	32.9	716	236
W 10 x 100	11.1	10.34	0.68	1.12	29.4	623	207
W 10 x 88	10.84	10.265	0.605	0.99	25.9	534	179
W 10 x 77	10.6	10.19	0.53	0.87	22.9	455	154
W 10 x 68	10.4	10.13	0.47	0.77	20	394	134
W 10 x 60	10.22	10.08	0.42	0.68	17.6	341	116
W 10 x 54	10.09	10.03	0.37	0.615	15.8	303	103
W 10 x 49	9.98	10	0.34	0.56	14.4	272	93.4
W 10 x 45	10.1	8.02	0.35	0.62	13.3	248	53.4
W 10 x 39	9.92	7.985	0.315	0.53	11.5	209	45
W 10 x 33	9.73	7.96	0.29	0.435	9.71	170	36.6
W 10 x 30	10.47	5.81	0.3	0.51	8.84	170	16.7
W 10 x 26	10.33	5.77	0.26	0.44	7.61	144	14.1
W 10 x 22	10.17	5.75	0.24	0.36	6.49	118	11.4
W 10 x 19	10.24	4.02	0.25	0.395	5.62	96.3	4.29
W 10 x 17	10.11	4.01	0.24	0.33	4.99	81.9	3.56
W 10 x 15	9.99	4	0.23	0.27	4.41	68.9	2.89
W 10 x 12	9.87	3.96	0.19	0.21	3.54	53.8	2.18
W 8 x 67	9	8.28	0.57	0.935	19.7	272	88.6
W 8 x 58	8.75	8.22	0.51	0.81	17.1	228	75.1
W 8 x 48	8.5	8.11	0.4	0.685	14.1	184	60.9
W 8 x 40	8.25	8.07	0.36	0.56	11.7	146	49.1
W 8 x 35	8.12	8.02	0.31	0.495	10.3	127	42.6
W 8 x 31	8	7.995	0.285	0.435	9.13	110	37.1
W 8 x 28	8.06	6.535	0.285	0.465	8.25	98	21.7
W 8 x 24	7.93	6.495	0.245	0.4	7.08	82.8	18.3
W 8 x 21	8.28	5.27	0.25	0.4	6.16	75.3	9.77
W 8 x 18	8.14	5.25	0.23	0.33	5.26	61.9	7.97
W 8 x 15	8.11	4.015	0.245	0.315	4.44	48	3.41
W 8 x 13	7.99	4	0.23	0.255	3.84	39.6	2.73
W 8 x 10	7.89	3.94	0.17	0.205	2.96	30.8	2.09
W 6 x 25	6.38	6.08	0.32	0.455	7.34	53.4	17.1
W 6 x 20	6.2	6.02	0.26	0.365	5.87	41.4	13.3

Table 8.21 RBDO with NN - Six-story space frame: American standard steel wide flange beam (W-shape) sections, Table 5 of 5.

Designation	Height (in)	Width (in)	Thickness web (in)	Thickness flange (in)	Area (in^2)	Moment of inertia I_y (in^4)	Moment of inertia I_z (in^4)
W 6 x 16	6.28	4.03	0.26	0.405	4.74	32.1	4.43
W 6 x 15	5.99	5.99	0.23	0.26	4.43	29.1	9.32
W 6 x 12	6.03	4	0.23	0.28	3.55	22.1	2.99
W 6 x 9	5.9	3.94	0.17	0.215	2.68	16.4	2.19
W 5 x 19	5.15	5.03	0.27	0.43	5.54	26.2	9.13
W 5 x 16	5.01	5	0.24	0.36	4.68	21.3	7.51
W 4 x 13	4.16	4.06	0.28	0.345	3.83	11.3	3.86

The deterministic constraints are eleven in total, two for the stresses of each element group and one for the inter-story drift. The load-displacement curve of a node in the top-floor of the frame is depicted in Figure 8.26, corresponding to the design vector (W 12×26, W 12×33, W 12×87, W 12×87, W 10×60) with a probability of failure equal to 8.68%. For this test example the $(\mu+\lambda)$-ES approach is used with $\mu=\lambda=5$ following the rule that μ, λ should be equal to the number of the design variables. A sample size of 500, 1000 and 5000 simulations have been examined for the MCS with the Importance Sampling technique (described in detail in Section 2.9), in order to study the influence of the number of simulations on the optimization process.

The probabilistic constraint is imposed on the allowable probability of structural collapse due to successive formation of plastic nodes and is set to $p_a=10^{-3}$. The probability of failure due to uncertainties related to material properties, geometry and loads of the structure is estimated using MCS with IS. The external load, the yield stresses, the elastic modulus and the dimensions of the cross-sections of the structural members are considered as random variables. The loads follow a log-normal Probability Density Function, while random variables associated with material properties and cross-section dimensions follow a normal PDF. The required importance sampling function $g_x(x)$ for the loads is also assumed to follow a normal distribution. The type of PDF, mean value, and variance of the random parameters are described in Table 8.22.

Table 8.22 RBDO with NN - Six-story space frame: Characteristics of the random variables.

Symbol (units)	Description	Probability density function	Mean value	Standard deviation
E (kN/m^2)	Young's Modulus	Normal	200	$0.10 \cdot E$
σ_y (kN/m^2)	Allowable stress	Normal	25.0	$0.10 \cdot \sigma_y$
x_i	Design variables	Normal	x_i	$0.1 \cdot x_i$
	Loads (×10^3)	Log- Normal	6.4	0.20

Table 8.23 shows the performance of the methods for the formulations considered. *DDO* stands for the conventional Deterministic Optimization approach, *RBDO* stands for the conventional Reliability-Based Design Optimization approach, while *RBDO-NNi* corresponds to the proposed Reliability-Based Optimization with NN incorporating *algorithm i* (i=1, 2).

Table 8.23 RBDO with NN - Six-story space frame: Performance of the methods.

Optimization procedure	no. of ES Generations	p_f**	Optimum weight (kN)	Time (h)
DDO	43	0.171	727	0.05
RBDO (500 siml.)	65	0.105×10^{-2}	869	7.6
RBDO-NN1 (500 siml.)	64	0.105×10^{-2}	873	2.7
RBDO-NN2 (500 siml.)	65	0.105×10^{-2}	869	3.6
RBDO (1000 siml.)	68	0.101×10^{-2}	875	16.3
RBDO-NN1 (1000 siml.)	69	0.101×10^{-2}	875	5.3
RBDO-NN2*	66	0.97×10^{-3}	881	5.0
RBDO (5000 siml.)	68	0.101×10^{-2}	875	81.1
RBDO-NN1 (5000 siml.)	69	0.101×10^{-2}	875	26.5
RBDO-NN2*	66	0.97×10^{-3}	881	5.0

*For 100000 simulations.
**For 100000 simulations using the NN2 scheme.

As can be observed from Table 8.23, the probability of failure for the deterministic optimum is unacceptable since it exceeds substantially the accepted threshold value of 10^{-3}. On the other hand, the optimum weight achieved by the RBDO exceeds by 16% the one obtained with the deterministic formulation. For the application of the RBDO-NN1 methodology the number of NN input units is equal to the number of design variables, whereas one output unit is needed. The output unit takes the values of 1 or 0, corresponding to a feasible or infeasible design vector, respectively. Consequently the NN configuration implemented in this case has one hidden layer with 10 nodes resulting in a 5-10-1 NN architecture used for all runs (Papadrakakis et al. 1996a). The training set consists of 100 training patterns chosen based on the requirement that the full range of the design space should be represented in the training procedure.

For the application of the RBDO-NN2 methodology the number of NN input units is equal to the number of the random variables, whereas one output unit is needed corresponding to the critical load factor. Consequently the NN configuration results in a 3-7-1 NN architecture which is used for all runs (Papadrakakis et al. 1996a). The number of conventional step-by-step limit state analysis calculations performed for the training of NN is taken 60 corresponding to different groups of random variables properly selected from the random field. As can be seen in Table 8.23 the proposed RBDO-NN1 and RBDO-NN2 optimization schemes for 500 and 1000 simulations manage to achieve the optimum weight in one third of the CPU time required by the conventional RBDO procedure. For 5000 simulations the time required for RBDO-NN1 remains one third (1/3),

while for the RBDO-NN2 the CPU time required drops to the one sixteenth (1/16) of the conventional times required. The number of MC simulations in the case of NN2 scheme can be practically unlimited without affecting the computational efficiency due to the trivial computing time required by the NN to perform one MCS. The difference on the computational time needed by the NN1 methodology, for different number of simulations, compared to NN2 is due to the fact that in the first methodology the computational time for the generation of the training set depends on the number of MC simulations which are required for checking the feasibility with respect to the probabilistic constraint. For this example it is observed that when the number of required simulations is less than 500, the RBDO-NN1 methodology outperforms the RBDO-NN2 methodology, the two methodologies perform similarly for 1000 simulations, while in the case of more than 5000 simulations the RBDO-NN2 needs one order of magnitude less computational time compared to both conventional and RBDO-NN1 methodologies. The computational time of the conventional RBDO is decreased by 70% when the RBDO-NN1 methodology is employed.

8.3.3 Conclusions on the two test examples of RBDO with NN

The aim of the proposed RBDO procedure of Sections 8.3.1 and 8.3.2 was threefold:

i. To reach an optimized design with controlled safety margins with regard to various model uncertainties;

ii. To minimize the weight of the structure;

iii. To reduce substantially the required computational effort.

The solution of realistic RBDO problems in structural mechanics is an extremely computationally intensive task. In the test examples considered the conventional RBDO procedure was found almost two orders of magnitude more computationally expensive than the corresponding deterministic optimization procedure. The aim of decreasing the computational cost by at least one order of magnitude was achieved using:

i. NN predictions to perform both deterministic and probabilistic constraints check;

ii. NN predictions to perform the structural analyses involved in MCS.

8.4 Reliability-based Robust Design Optimization (RRDO)

In this section two Reliability-based Robust Design Optimization test examples are considered, namely a 39-bar space truss and a space truss tower. The objective functions considered are the initial construction cost and the variance of the response of the structure, while each design is checked whether it satisfies the provisions of the European design codes for steel structures (Eurocode 3) (CEN 1993) with a prescribed probability of violation. The uncertainty of loads, material properties and cross section dimensions is taken into consideration and reliability analysis is performed with the Monte Carlo Si-

mulation method, described in Section 2.6, combined with the Latin Hypercube Sampling technique of Section 2.8.

The optimization problem is handled with the non-dominant Cascade Evolutionary Algorithm with the weighted Tchebycheff metric, described in Section 4.10.1. In order to make the application of this robust and efficient optimization procedure feasible to real-world problems, Neural Networks are implemented to replace the conventional finite element analyses required by the MCS.

8.4.1 39-bar space truss – RRDO test example

The test example considered is the 39-bar truss structure shown in Figure 8.4 (Lagaros et al. 2007), the same structure used for the test example of Section 8.1.2. The height of the structure is 16 m, while its basis is an equilateral triangle of side 6.93m. The model consists of 15 nodes and 39 elements divided into 4 design variables. The design variables considered are the dimensions of the members of the structure, four groups in total, taken from the Circular Hollow Section table of the Eurocode (CEN 1993), shown in Table 8.3.

For each design variable, two stochastic variables are assigned, the external diameter d and the thickness t of the circular hollow section. A vertical load V=2kN is applied to all nodes, while a probabilistic horizontal load F with a mean value 8kN is applied to the top nodes at the x-direction. The type of Probability Density Function, the mean value, and the variance of the random variables are shown in Table 8.5.

Three types of constraints are imposed to the sizing optimization problem: (i) stress; (ii) compression force (for buckling); and (iii) displacement constraints. The constraints are the same as the ones used for the numerical application of Section 8.1.2. Stress constraints are described in Eqs. (8.1) and (8.2), buckling constraints are described in Eqs. (8.3), (8.4) and (8.5), while the displacement constraint is described in Eq. (8.6). For this test example, a constraint of 500mm on the maximum deflection is imposed, as opposed to the corresponding constraint imposed for the test example of Section 8.1.2, which was 200mm.

Parametric study for MCS with LHS

In order to examine the performance of the MCS combined with LHS, a parametric study is performed for monitoring the influence of the number of LHS simulations with respect to the accuracy that the required statistical quantities are calculated. For this reason the probability of violation of displacement constraint and the standard deviation of the characteristic node displacement are calculated with respect to the number of LHS simulations for a randomly selected design V. The properties of the specific design V are shown in Table 8.24.

Table 8.24 RRDO - 39-bar space truss: Properties of design *V* for verification.

Property	Value
Section 1 (*D*×*t*)*	114.3 × 6.3
Section 2 (*D*×*t*)*	88.9 × 8
Section 3 (*D*×*t*)*	101.6 × 8
Section 4 (*D*×*t*)*	114.3 × 6.3
Weight (kN)	107.91
Standard Deviation	0.14
p_{viol} (%)	2.40

* Taken from the Circular Hollow Section table of the Eurocode

The results of the parametric study for this design can be seen in Figure 8.27. For this test example 3000 LHS simulations have been considered adequate in order to calculate with sufficient accuracy the statistical quantities under consideration.

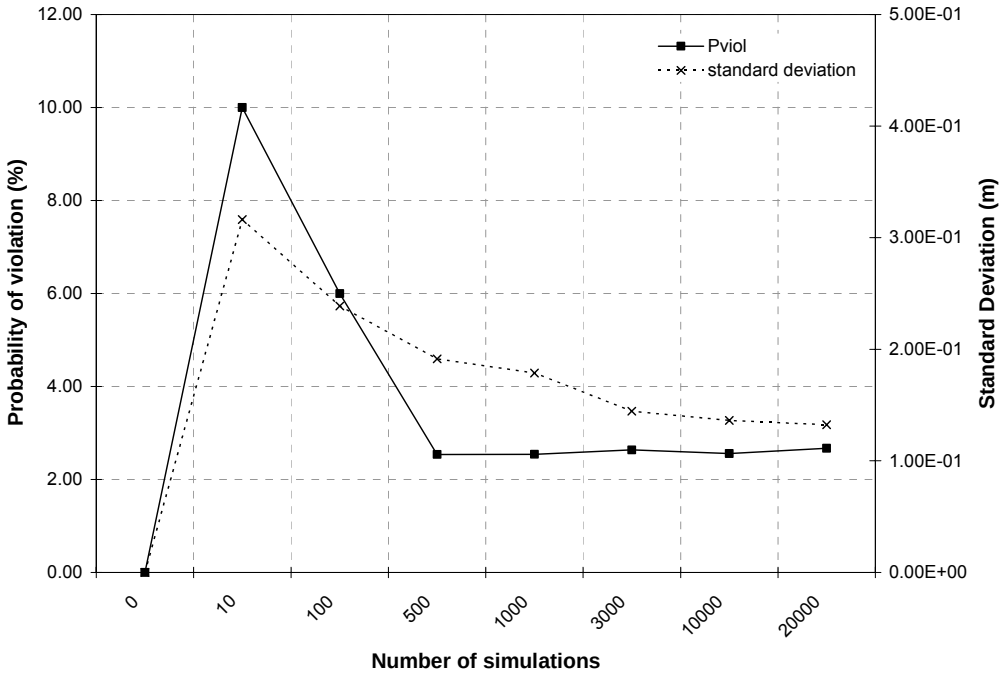


Figure 8.27 RRDO - 39-bar space truss: Verification for design *V*.

Solution of the RDO problem

For the solution of the multi-objective optimization problem in question the non-dominant $CEATm(\mu+\lambda)_{nrun,csteps}$ optimization scheme was employed where $\mu=\lambda=5$, $nrun=10$ and $csteps=3$. These values have been found to be appropriate for providing good quality Pareto front curves (Lagaros et al. 2005c). Two different formulations of the RDO problem have been considered in this study:

i. The standard RDO formulation;

ii. The RRDO formulations with probabilistic constraints.

In the case where probabilistic constraints are taken into account, a probability of violation equal to 2% has been considered. The resultant Pareto front curves for the two RDO formulations are depicted in Figure 8.28, with the structural weight on the horizontal axis and the standard deviation of the characteristic node displacement on the vertical axis.

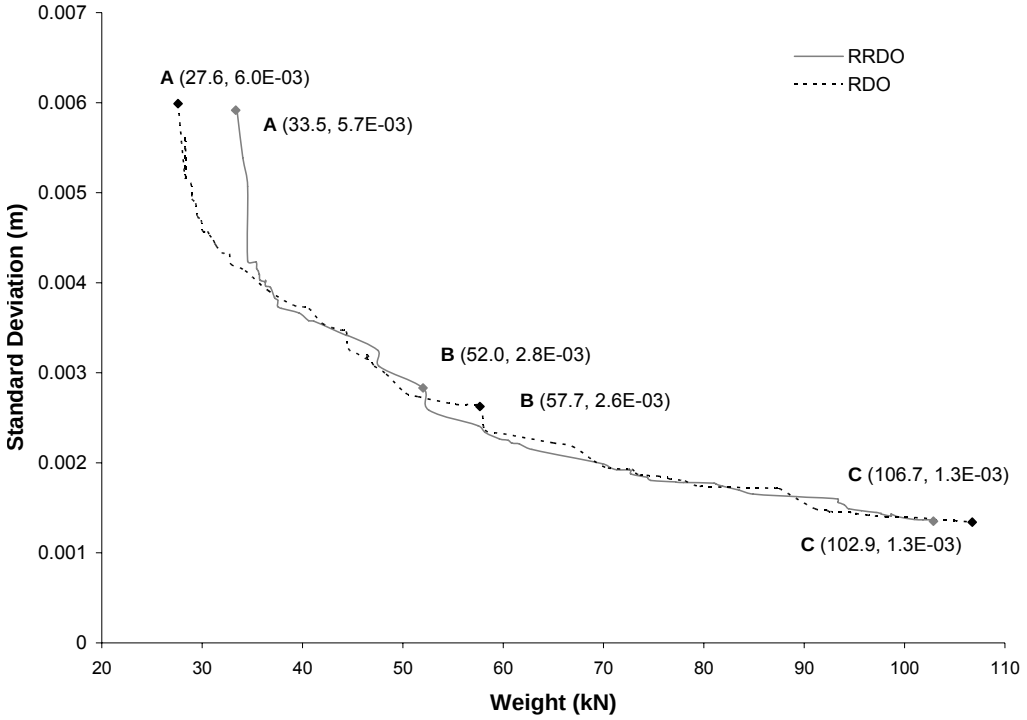


Figure 8.28 RRDO - 39-bar space truss: Comparison of the Pareto front curves.

It can be seen from Figure 8.28 that the influence of considering the probabilistic constraints is significant when the weight of the structure is the dominant criterion (designs A_{RDO} and A_{RRDO}). The two Pareto front curves almost coincide when the importance of the second criterion (standard deviation of the response) becomes dominant.

Table 8.25 RRDO - 39-bar space truss: Characteristic optimal solutions.

	RDO			RRDO		
	DDO (*A*)	**(*B*)**	**(*C*)**	**DDO (*A*)**	**(*B*)**	**(*C*)**
Section 1 (*D*×*t*)*	139.7 × 4	168.3 × 6.3	193.7 × 12.5	139.7 × 5	139.7 × 10	193.7 × 12.5
Section 2 (*D*×*t*)*	139.7 × 5	219.1 × 7.1	193.7 × 12.5	168.3 × 4.5	114.3 × 10	193.7 × 12.5
Section 3 (*D*×*t*)*	101 × 6.3	168.3 × 8	193.7 × 12.5	101.6 × 10	139.7 × 8	193.7 × 12.5
Section 4 (*D*×*t*)*	139.7 × 4	193.7 × 6.3	193.7 × 12.5	139.7 × 4	101.6 × 10	273 × 7.1
Weight (kN)	27.6	57.7	106.7	33.5	52.0	102.9
Variance (m)	6.04×10^{-3}	2.62×10^{-3}	1.34×10^{-3}	5.71×10^{-3}	2.83×10^{-3}	1.35×10^{-3}
p_{viol} (%)	4.2	3.9×10^{-1}	9.0×10^{-3}	1.9	5.1×10^{-1}	9.0×10^{-3}

* Taken from the Circular Hollow Section table of the Eurocode.

In Table 8.25 three designs are compared which have been selected from the two Pareto front curves. Designs B_{RRDO} versus B_{RDO} and C_{RRDO} versus C_{RDO} are similar, with respect to both weight and standard deviation of the response, leading to similar probabilities of violation. On the other hand designs A_{RRDO} and A_{RDO} differ by 17.5% with respect to the weight and by 6.0% with respect to the standard deviation of the characteristic node displacement. Moreover the probability of violation of the constraints, in the case of the A_{RDO} design, is equal to 4.2% while it becomes equal to 1.9% for the A_{RRDO} design.

8.4.2 Truss tower – RRDO test example

The test example considered is the 3D truss tower shown in Figures 8.29 and 8.30 (Lagaros et al. 2007). The height of the truss tower is 128 m, while its basis is a rectangle of side 17.07 m. The model consists of 324 nodes and 1254 elements divided into 12 groups that play the role of the design variables. The design variables considered are the dimensions of the members of the structure, 12 groups in total, taken from the Circular Hollow Section table of the Eurocode (CEN 1993), shown in Table 8.3. For each design variable, two stochastic variables are assigned, the external diameter d and the thickness t of the circular hollow section. The type of PDFs, the mean value, and the variance of the random variables are shown in Table 8.26.

Three types of constraints are imposed to the sizing optimization problem: (i) stress; (ii) compression force (for buckling); and (iii) displacement constraints. The constraints are the same as the ones used for the numerical application of Section 8.1.2. Stress constraints are described in Eqs. (8.1) and (8.2), buckling constraints are described in Eqs. (8.3), (8.4)and (8.5), while the displacement constraint is described in Eq. (8.6). For this test example, a constraint of 500mm on the maximum deflection is imposed.

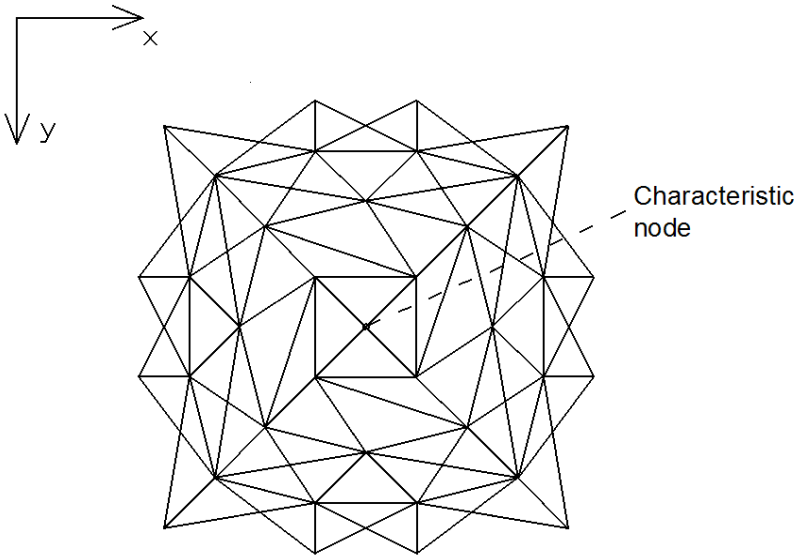


Figure 8.29 RRDO - Truss tower: Top view of the structure.

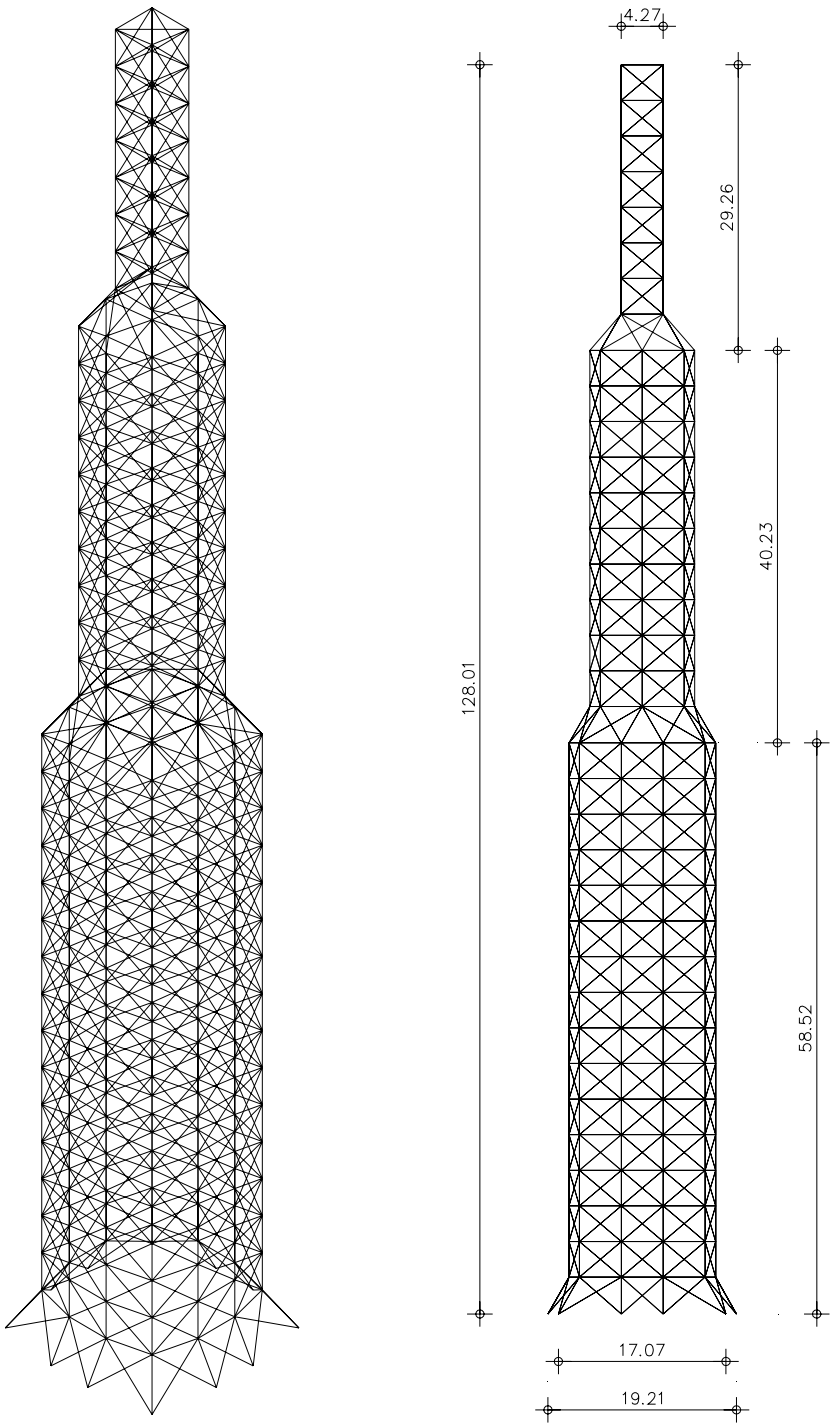


Figure 8.30 RRDO - Truss tower: Views of the structure:
(a) 3D view, (b) Front view (dimensions in m).

Table 8.26 RRDO - Truss tower: Characteristics of the random variables.

Symbol (units)	Description	PDF	Mean value	Coefficient of variation (%)
E (kN/m^2)	Young's Modulus	Normal	2.10×10^8	7.14
σ_y (kN/m^2)	Allowable stress	Normal	355000	10.00
F (kN/m^2)	Wind loading	Normal	F_μ	40.00
D	CHS Diameter	Normal	D_i *	2.0
t	CHS Thickness	Normal	t_i *	2.0

* Taken from the Circular Hollow Section table of the Eurocode, for every design

The applied loading consists of:

i. Self weight (dead load);

ii. Live loads;

iii. Wind actions according to the Eurocode (CEN 1991).

Parametric study for MCS with LHS

A similar to the previous test example parametric study, is also performed for this test example, in order to examine the applicability and the efficiency of MCS with LHS. The probability of violation of the displacement constraint and the standard deviation of the characteristic node displacement are calculated with respect to the number of LHS simulations for a randomly selected design V. The properties of the specific design V are shown in Table 8.27.

Table 8.27 RRDO - Truss tower: Properties of design V for verification.

Property	Value
Section 1 ($D\times t$)*	114.3 × 6.3
Section 2 ($D\times t$)*	88.9 × 8
Section 3 ($D\times t$)*	101.6 × 8
Section 4 ($D\times t$)*	114.3 × 6.3
Volume (m^3)	107.91
Standard Deviation (m)	0.26
p_{viol} (%)	6.80

* Taken from the Circular Hollow Section table of the Eurocode.

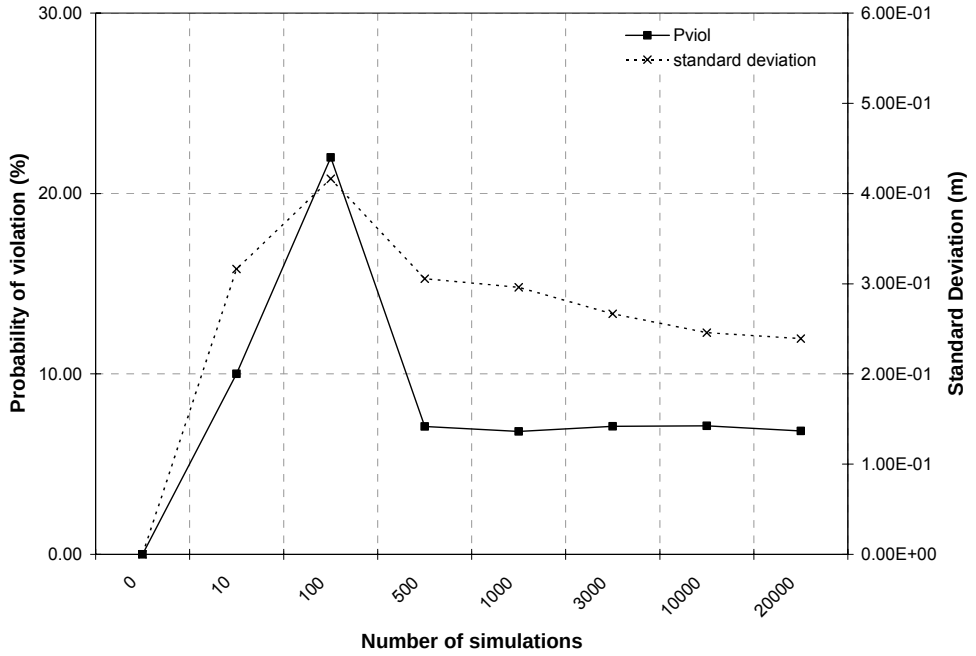


Figure 8.31 RRDO - Truss tower: Verification.

The results of the parametric study for the specific design V are shown in Figure 8.31. For this test example 3000 LHS simulations have been considered enough in order to calculate with sufficient accuracy the statistical quantities under consideration.

Solution of the RDO problem

For the solution of the multi-objective optimization problem in question the non-dominant CEATm$(\mu+\lambda)_{nrun,csteps}$ optimization scheme was employed where $\mu=\lambda=5$, $nrun=10$ and $csteps=3$. The resultant Pareto front curves for the two RDO formulations are depicted in Figure 8.32.

As can be seen from Figure 8.32, the trend on the influence of the probabilistic constraints is similar to the one of the previous test example, i.e. it is significant near the area where the weight of the structure is the dominant criterion. As the importance of the standard deviation of the response increases, the two Pareto front curves almost coincide.

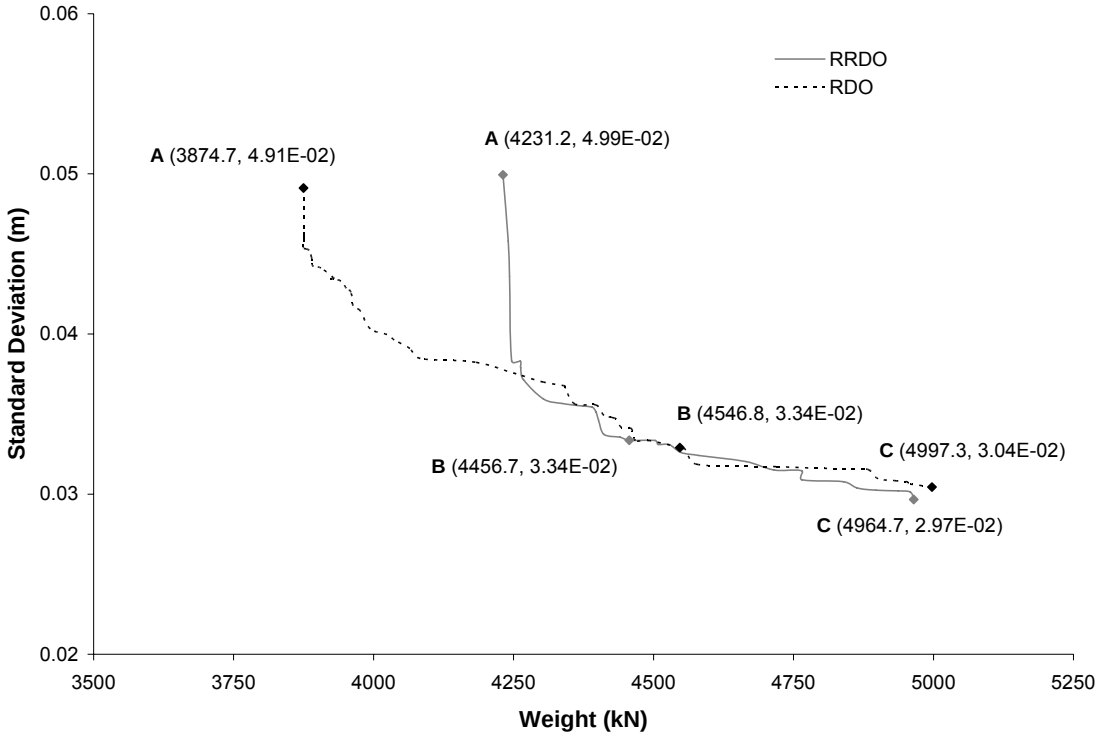


Figure 8.32 RRDO - Truss tower: Comparison of the Pareto front curves.

Table 8.28 RRDO - Truss tower: Characteristic optimal solutions.

	RDO			RRDO		
	DDO (*A*)	**(*B*)**	**(*C*)**	**DDO (*A*)**	**(*B*)**	**(*C*)**
Section 1 (*D*×*t*)*	323.9 × 8.0	355.6 × 10.0	406.4 × 8.8	323.9 × 8.0	323.9 × 8.0	323.9 × 8.0
Section 2 (*D*×*t*)*	355.6 × 8.0	219.1 × 20.0	406.4 × 10.0	323.9 × 8.0	323.9 × 8.0	323.9 × 12.5
Section 3 (*D*×*t*)*	355.6 × 10.0	355.6 × 10.0	406.4 × 8.8	355.6 × 10.0	355.6 × 10.0	323.9 × 12.5
Section 4 (*D*×*t*)*	323.9 × 10.0	244.5 × 16.0	323.9 × 12.5	355.6 × 10.0	219.1 × 20.0	406.4 × 8.8
Section 5 (*D*×*t*)*	406.4 × 8.8	406.4 × 8.8	406.4 × 8.8	406.4 × 8.8	323.9 × 12.5	406.4 × 8.8
Section 6 (*D*×*t*)*	355.6 × 8.0	355.6 × 8.0	355.6 × 10.0	355.6 × 8.0	355.6 × 8.0	273.0 × 12.5
Section 7 (*D*×*t*)*	323.9 × 8.0	273.0 × 8.0	323.9 × 10.0	323.9 × 8.0	323.9 × 8.0	219.1 × 16.0
Section 8 (*D*×*t*)*	323.9 × 10.0	323.9 × 12.5	244.5 × 20.0	355.6 × 10.0	355.6 × 10.0	273.0 × 16.0
Section 9 (*D*×*t*)*	323.9 × 8.0	323.9 × 8.0	355.6 × 8.0	323.9 × 8.0	323.9 × 8.0	323.9 × 8.0
Section 10 (*D*×*t*)*	219.1 × 7.1	323.9 × 8.0	273.0 × 12.5	355.6 × 8.0	355.6 × 8.0	219.1 × 20.0
Section 11 (*D*×*t*)*	323.9 × 10.0	244.5 × 20.0	244.5 × 20.0	406.4 × 8.8	244.5 × 20.0	244.5 × 20.0
Section 12 (*D*×*t*)*	323.9 × 8.0	323.9 × 12.5	273.0 × 16.0	323.9 × 10.0	273.0 × 12.5	273.0 × 16.0
Weight (kN)	3874.7	4546.8	4997.3	4231.2	4456.7	4964.7
Variance (m)	4.91×10^{-2}	3.34×10^{-2}	3.04×10^{-2}	4.99×10^{-2}	3.34×10^{-2}	2.97×10^{-2}
p_{viol} (%)	3.80	2.80×10^{-1}	8.5×10^{-3}	1.80	2.90×10^{-1}	8.0×10^{-3}

* Taken from the Circular Hollow Section table of the Eurocode.

In Table 8.28 three designs are compared which have been selected from the two Pareto front curves. Designs B_{RRDO} versus B_{RDO} and C_{RRDO} versus C_{RDO} are similar with respect to both weight and standard deviation of the response while they have similar probabilities

of violation. On the other hand designs A_{RRDO} and A_{RDO} differ by 10.0% with respect to the weight and by 2.0% with respect to the standard deviation of the characteristic node displacement. Moreover the probability of violation of the constraints, in the case of the A_{RDO} design, is equal to 3.8% while it becomes equal to 1.8% for the A_{RRDO} design.

8.4.3 Conclusions on the two test examples of RRDO

The proposed methodology is tested and validated in the two test examples of Sections 8.4.1 and 8.4.2. For both test examples, two objective functions have been considered: the construction cost and the standard deviation of a characteristic node displacement, representative of the structural response under the external loading. Two sets of constraints are enforced: deterministic constraints on stresses; element buckling and displacements imposed by the Eurocode 3 (CEN 1993), and a probabilistic constraint on the maximum probability of violation of the deterministic constraints. Furthermore, due to manufacturing limitations the design variables are not considered continuous but discrete. The discrete design variables are treated in the same way as in single objective design optimization problems using the discrete version of Evolution Strategies (Papadrakakis et al. 1998a). The design variables considered are the dimensions of the members of the structure taken from the Circular Hollow Section table of the Eurocode (CEN 1993). The random variables related to the cross-sectional dimensions, for both test examples, are two per design variable: the external diameter D and the thickness t of the circular hollow section. Apart from the cross-sectional dimensions of the structural members, the material properties (modulus of elasticity E and yield stress σ_y) as well as the lateral loads have been considered as random variables.

8.5 Reliability-based Robust Design Optimization (RRDO) assisted by Neural Networks

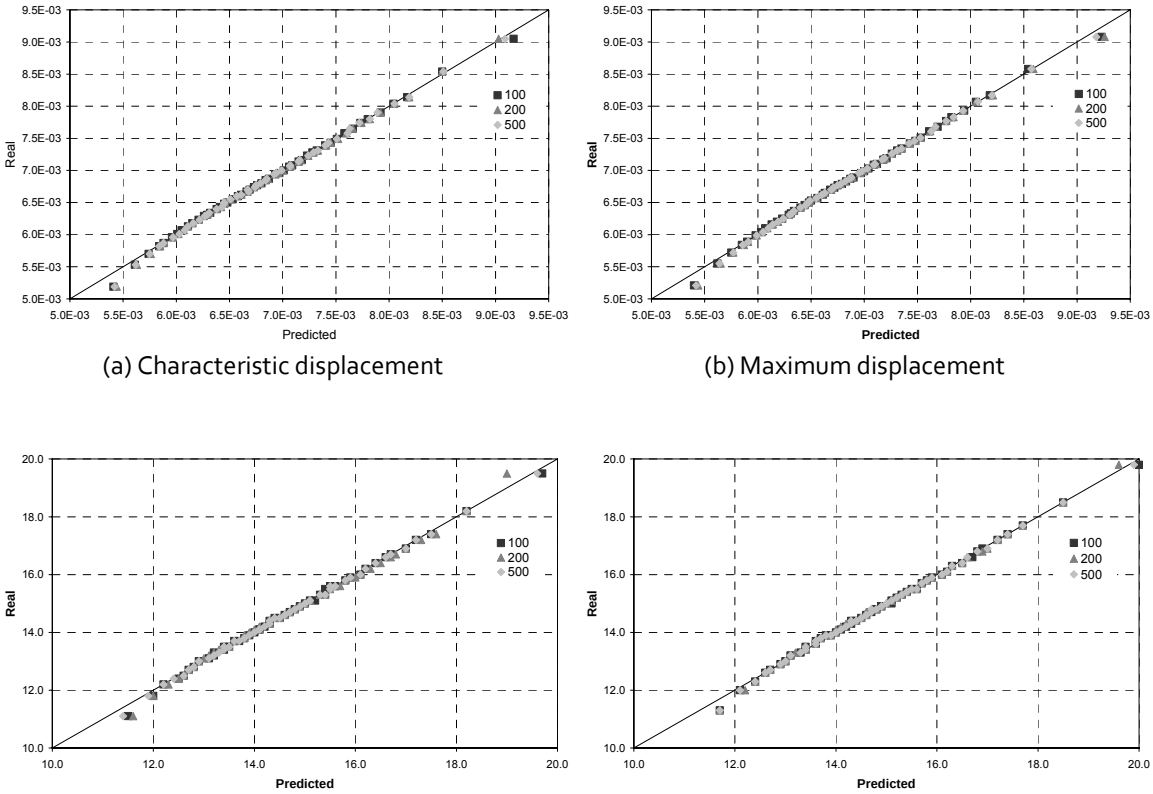
8.5.1 39-bar truss – RRDO test example with NN

The first test example considered is the 39-bar truss structure of the test example of Section 8.4.1. All properties and parameters of this test example are the same as the ones used for the test example of Section 8.4.1, with the only difference of incorporating NN predictions instead of conventional methods.

The NN configuration implemented has one hidden layer with 30 nodes resulting in a fully connected 11-30-4 NN architecture which is used for all runs. The 11 input nodes of the NN correspond to the random variables of Table 8.5, which are the dimensions D and t of the four design variables, the modulus of elasticity, the allowable stress of the structural members and the value of the horizontal load. On the other hand, the four output nodes of the network correspond to the characteristic displacement, the maximum displacement, the maximum compressive force and the maximum tensile force.

Study on the performance of NN predictions

The performance of the NN predictions is demonstrated in Figure 8.33 for a random design *A*, shown in Table 8.29.



(a) Characteristic displacement (b) Maximum displacement

(c) Maximum compressive force (d) Maximum tensile force

Figure 8.33 RRDO - 39-bar space truss with NN: Performance of NN with respect to the number of the training patterns (for design *A*, shown in Table 8.29).

Table 8.29 RRDO - 39-bar space truss with NN:
Design *A*, to be used for checking the performance of NN.

Design variable	Cross section
Section 1	139.7 x 12.5
Section 2	193.7 x 16
Section 3	193.7 x 10
Section 4	168.3 x 10

Three different training sets, of size 100, 200 and 500 have been examined, all generated randomly using the LHS method, while 50 patterns (also randomly generated with LHS) have been used for testing the prediction capabilities of the trained NN over the entire range of interest. Due to the special care taken to alleviate the possibility of NN extrapolation (see Section 6.7.1), as can be seen in Figure 8.33, 100 patterns are adequate for training efficiently the NN. In order to assess the accuracy of the NN predictions, the

probability of violation was estimated using the same NN architecture trained with 100, 200 and 500 patterns and compared to the "exact" values as obtained by the conventional LHS-MCS. The results are shown in Table 8.30 for various numbers of MC simulations where it can be seen that the actual probability of violation is 0.53%, obtained after 5000 MC simulations. It is confirmed that 100 training patterns are adequate for an accurate prediction since a probability of violation equal to 0.54% is obtained by the NN.

Table 8.30 RRDO - 39-bar space truss with NN: Accuracy study of the NN predictions for a training set of 100, 200 and 500 patterns.

	Probability of violation (%)			
No. of Monte Carlo Simulations	**"Exact"**	**NN-100 patterns**	**NN-200 patterns**	**NN-500 patterns**
500	1.00	1.40	1.20	1.40
1000	7.00×10^{-1}	9.00×10^{-1}	8.00×10^{-1}	9.00×10^{-1}
2000	5.50×10^{-1}	6.50×10^{-1}	6.00×10^{-1}	6.50×10^{-1}
3000	5.00×10^{-1}	5.67×10^{-1}	5.33×10^{-1}	6.33×10^{-1}
5000	5.30×10^{-1}	5.40×10^{-1}	5.20×10^{-1}	5.20×10^{-1}
10000	5.30×10^{-1}	5.40×10^{-1}	5.20×10^{-1}	5.20×10^{-1}

For the solution of the multi-objective optimization problem in question the non-dominant CEATm$(\mu+\lambda)_{nrun,csteps}$ optimization scheme was implemented with $\mu=\lambda=5$, $nrun=10$ and $csteps=3$. These values were found to be suitable for providing good quality Pareto front curves (Lagaros et al. 2005c). Four different formulations of the RDO problem have been considered:

1. The standard RDO formulation;
2. The RRDO_2% with allowable probability of violation of the deterministic constraints equal to 2%;
3. The RRDO_0.1% with allowable probability of violation equal to 0.1%;
4. The RRDO_0.01% with allowable probability of violation equal to 0.01%.

The resultant Pareto front curves for the above mentioned formulations are depicted in Figure 8.34, with the construction cost on the horizontal axis and the standard deviation of the characteristic node displacement on the vertical axis.

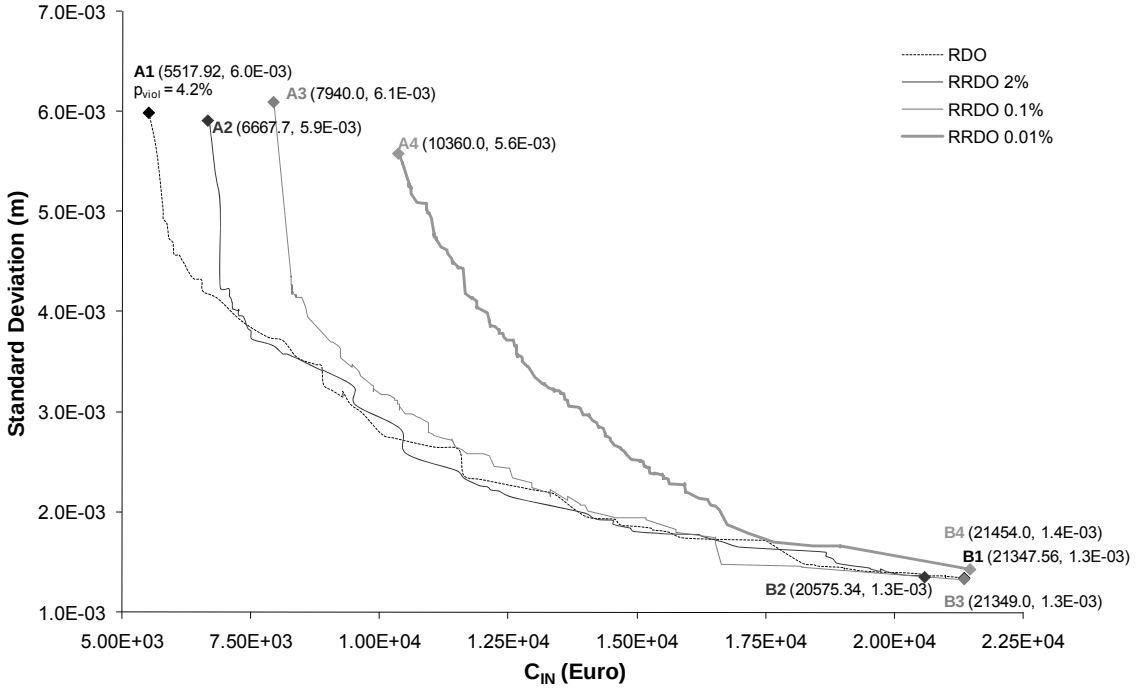


Figure 8.34 RRDO - 39-bar space truss with NN: Comparison of the Pareto front curves.

The Pareto front curve shown in Figure 8.34 represents solutions favoring one or the other objective function. Solutions near optimal design points *A* favor the objective function that is related to the initial cost, while solutions near the design points *B* favor the objective function that is related to the standard deviation of the response. All other intermediate points are compromise solutions between these two optimal solutions that are conflicting to each other.

It can be seen in Figure 8.34 that the influence of the probabilistic constraint is significant near the optimum designs (A_i, i=1, ...,4). These designs are located in the Pareto front area dominated by the initial construction cost (C_{IN}) of the structure, where the four designs are substantially affected by the conditions imposed by the probabilistic constraint. It can furthermore be seen that the four Pareto front curves gradually converge as the importance of the second criterion (standard deviation of the response), increases. In particular, the Pareto fronts, obtained with the RDO and RRDO_2% formulations, approach each other first, followed by the Pareto front of RRDO_0.1%, while all four Pareto fronts converge at the lower end of the design space.

The optimization formulations considered require different sample sizes for the MCS. For RDO and RRDO_2% formulations, a sample size of 10 000 was adequate, while for RRDO_0.1% and RRDO_0.01% formulations, a sample size of 100 000 and 500 000 was required, respectively. Different formulations and consequently different sample sizes lead to significantly different computing costs. The computational effort required for solving the optimization problem is given in Table 8.31, where the conventional and the NN-based computing times are reported. It can be seen that the computational time is

reduced by three orders of magnitude for RRDO_0.1% and RRDO_0.01% cases when NN predictions are implemented.

Table 8.31 RRDO - 39-bar space truss with NN: Computational efficiency study.

Formulation	No. of Simulations	Time (h)	
		LHS-MCS	MCS-NN
RDO	10000	1.81	2.17×10^{-2}
RRDO 2%	10000	1.82	2.16×10^{-2}
RRDO 0.1%	100000	1.76×10^{1}	2.12×10^{-2}
RRDO 0.01%	500000	8.60×10^{1}	2.06×10^{-2}

8.5.2 Truss tower – RRDO test example with NN

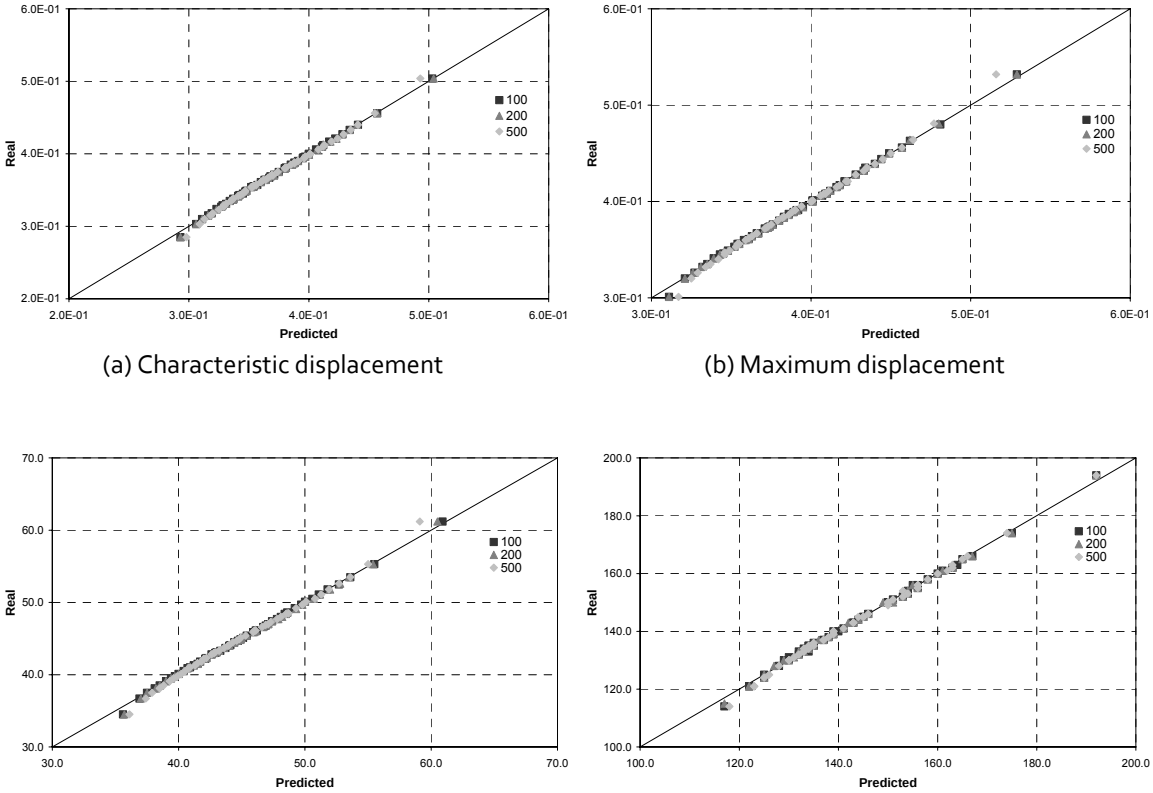
The second test example considered is the 3D truss tower of the test example of Section 8.4.2. All properties and parameters of this test example are the same as the ones used for the test example of Section 8.4.2, with the only difference of implementing NN predictions instead of conventional analyses. This test example is selected in order to demonstrate the performance, in terms of accuracy and computational efficiency, of the proposed optimization scheme for a rather large-scale optimization problem.

The FE model consists of 324 nodes and 1254 elements which are divided into 12 groups of design variables. The applied loading consists of: (i) self weight (dead load); (ii) live loads; and (iii) wind actions according to the Eurocode 1 (CEN 1991). The type of PDFs, the mean value, and the variance of the random variables are shown in Table 8.26. The horizontal displacement (in the x-direction) of the top node is selected as the characteristic displacement to be monitored, as shown in Figure 8.29.

The NN configuration implemented has one hidden layer with 50 nodes resulting in a fully connected 27-50-4 NN architecture. The 27 input nodes of the NN correspond to the random variables of Table 8.26, which are the dimensions D and t of the twelve design variables, the modulus of elasticity, the allowable stress of the structural members and the value of the horizontal load.

Study on the performance of NN predictions

The performance of NN is demonstrated in Figure 8.35, where the prediction of the characteristic displacement, the maximum displacement, the maximum compressive force and the maximum tensile force are shown for a random design A, shown in Table 8.32.



(a) Characteristic displacement (b) Maximum displacement

(c) Maximum compressive force (d) Maximum tensile force

Figure 8.35 RRDO - Truss tower with NN: Performance of NN with respect to the number of the training patterns (for design *B*, shown in Table 8.32).

Table 8.32 RRDO - Truss tower with NN: Design *B*, to be used for checking the performance of NN.

Design variable	Cross section
Section 1	323.9 x 16
Section 2	193.7 x 10
Section 3	273 x 20
Section 4	219.1 x 16
Section 5	323.9 x 7.1
Section 6	323.9 x 16
Section 7	244.5 x 20
Section 8	273 x 16
Section 9	273 x 10
Section 10	244.5 x 16
Section 11	273 x 7.1
Section 12	323.9 x 8.0

Three different training sets, of 100, 200 and 500 sample size have been examined, all generated randomly using the LHS method. As can be seen, 100 patterns are adequate for

training efficiently the NN. In accordance to the previous test example, in order to test the accuracy of the NN predictions, the probability of violation is also assessed for the three patterns sizes. The obtained results are shown in Table 8.33 where it can be seen that 500 patterns give a more accurate prediction of the probability of violation compared to 100 or 200 patterns. The computing cost is given in Table 8.34, where the computing times of the conventional and the NN-based optimization procedures are reported. It has to be noted that the computing costs of the conventional RRDO_0.1% and RRDO_0.01% formulations are estimations due to the excessive computing cost required for these two cases. It can be seen that the NN-based methodology requires up to three orders of magnitude less computing time compared to the conventional formulations. The optimization scheme used in the previous test example is also implemented here and the resultant Pareto front curves for the RDO formulations are depicted in Figure 8.36 with and without probabilistic constraints. Similar behavior to the previous test example can be observed regarding the quality of the Pareto front curves and the dominant regions of each objective function.

Table 8.33 RRDO - Truss tower with NN: Accuracy study of the NN predictions for a training set of 100, 200 and 500 patterns.

	Probability of violation (%)			
No. of Monte Carlo Simulations	**"Exact"**	**NN-100 patterns**	**NN-200 patterns**	**NN-500 patterns**
500	0.00	0.00	0.00	0.00
1000	1.00×10^{-1}	2.00×10^{-1}	2.00×10^{-1}	1.00×10^{-1}
2000	2.00×10^{-1}	2.50×10^{-1}	2.50×10^{-1}	2.00×10^{-1}
3000	2.00×10^{-1}	2.33×10^{-1}	2.33×10^{-1}	2.00×10^{-1}
5000	2.22×10^{-1}	2.67×10^{-1}	2.89×10^{-1}	2.44×10^{-1}
10000	2.22×10^{-1}	2.67×10^{-1}	2.89×10^{-1}	2.44×10^{-1}

Table 8.34 RRDO - Truss tower with NN: Computational efficiency study.

Formulation	**No. of Simulations**	**Time (hours)**	
		LHS-MCS	**MCS-NN**
RDO	10000	53.3	3.08
RRDO 2%	10000	54.2	3.12
RRDO 0.1%	100000	559 *	3.19
RRDO 0.01%	500000	2790 *	3.50

* Estimated.

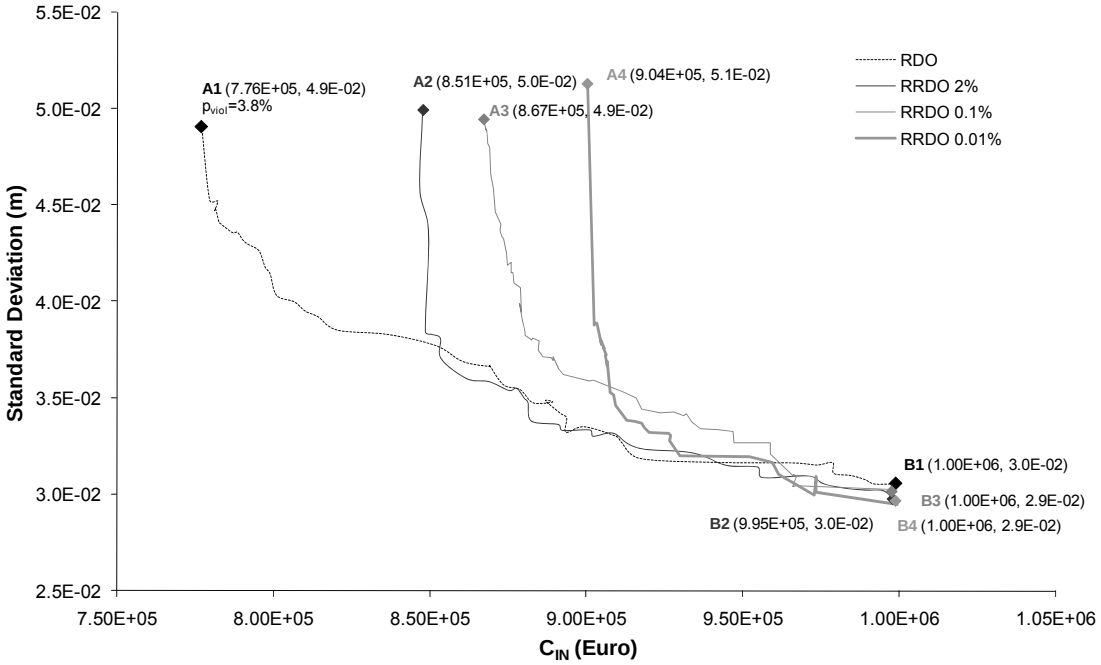


Figure 8.36 RRDO - Truss tower with NN: Comparison of the Pareto front curves.

8.5.3 Conclusions on the two test examples of RRDO with NN

The deterministic optimum is not always a "safe" design, since there are many random factors that affect the design during its lifetime. In order to find the "real" optimum the designer has to take into account all important random parameters. For this purpose two separate formulations have been performed in the past: the reliability-based optimization and the robust design optimization. However, due to the excessive computational demands required for addressing these type of problems, all studies were restricted to very simple academic examples. In the present work a reliability analysis combined with a robust design optimization formulation is proposed where probabilistic constraints are incorporated into the formulation of the robust design optimization problem.

The aim of this work in addressing a structural optimization problem considering uncertainties was twofold: (i) In the first part, the influence of the probabilistic constraints was examined, and in particular those related to the probability of violation of behavioral constraints, in addition to the variance of the structural response due to a number of uncertainties; (ii) In the second part, a neural network-based methodology was implemented for accelerating the computational efficiency of the proposed methodology and making the application of this type of optimization problems feasible for realistic structures.

The combined RRDO formulation requires structural reliability analyses to be performed for every candidate design of the ES-based optimizer, in order to determine the maximum probability of violation of the constraints. Depending on the value of the allowable probability of violation, different sample sizes are used in order to calculate with an ac-

ceptable accuracy the statistical quantities under consideration, i.e. the standard deviation of the structural response and the probability of violation of the constraints. The Pareto front curves obtained for the combined reliability-based robust design optimization and the robust design optimization formulations are much different when the initial construction cost is predominant in the objective function. However, when the standard deviation criterion becomes dominant, the Pareto front curves for various probabilities of violation almost coincide. Consequently, the optimum initial construction costs achieved by the combined reliability-based robust design optimization formulations are larger than the corresponding initial construction cost achieved by the robust design optimization. Furthermore, it was observed that an increasing influence of the standard deviation in the objective function, forces the robust optimum design to produce results very close to those obtained by the combined reliability-based robust design optimization formulations close to the right end of the Pareto front curve.

It has been also demonstrated that the solution of a combined RRDO of real-world problems in structural mechanics is an extremely computationally intensive task. Although the Latin Hypercube Sampling methodology has been implemented for improving the computational efficiency of the Monte Carlo Simultion method, the computational cost remains excessive making the solution of such problems computationally unsolvable. In this work, a neural network assisted methodology has been proposed in order to obtain estimates of the structural response required during the reliability analysis with the Monte Carlo Simulation method. The obtained NN estimates have shown to be very satisfactory in terms of accuracy for performing this type of computations. Furthermore, the present numerical results manage to achieve a reduction in computational time up to four orders of magnitude, for low probabilities of violation, compared to the conventional procedure making thus feasible the reliability-based robust design optimization of realistic structures under probabilistic constraints.

Chapter 9

9 Conclusions

9.1 Original contribution of the thesis

The objective of the present thesis was to develop algorithms and methodologies for solving the problem of structural design optimization considering uncertainties. The problem can be stated in various forms, such as *Reliability-Based Design Optimization* (RBDO), *Robust Design Optimization* or the combined *Reliability-based Robust Design Optimization* (RRDO) problem. These problems have some unique characteristics and require a set of different algorithms and techniques in order to be solved:

i. For the basic optimization process, a single-objective optimization algorithm is required. The algorithm should be robust and efficient, capable of finding the global optimum, without being trapped in local optima, with a satisfactory convergence rate and consequently not requiring excessive computational effort.

ii. For the stochastic analysis part of the methodology, a technique is required that can calculate the statistical quantities that are affected by the random variables of the model. These quantities can be probabilities, such as the probability of failure of the structure or the probability of violation of the behavioral constraints of the optimization problem, or other statistical parameters (mean value, standard deviation, etc) of derived random quantities.

iii. For the multi-objective optimization problem encountered in the RDO or in the RBDO formulations considering multiple objectives, a multi-objective optimization algorithm is required. The algorithm should be able to provide a well distributed Pareto Front, without being trapped in local optima, even when non-convex functions are describing the optimization problem.

iv. Due to the fact that the problem of optimum structural design considering uncertainties, in any one of its formulations, is extremely computationally intensive, especially when real-world large-scale structures with many design variables and/or random variables are considered, it is required to implement some special tech-

niques, such as metamodels, that can help in substantially reducing the computational effort of the whole process.

The present thesis contributed to each one of the four points mentioned above, as will be described in detail in the next sections.

9.1.1 Single-objective optimization

The RBDO, RDO and RRDO problems are computationally very expensive. Consequently, the single-objective optimizer used for these problems should be efficient and robust, without requiring excessive computational effort in order to locate the optimum. Furthermore, it should be a global optimizer, capable of finding the global optimum, without being trapped in the various local optima that might exist in the design space. In the present thesis, *Evolutionary Algorithms* (EAs) were mainly used and in particular the *Evolution Strategies* (ES) algorithm. A first version of the algorithm for structural optimization was implemented by Papadrakakis et al. (1998a). In the present thesis, the algorithm was used in its discrete and continuous forms and was fine-tuned in order to be robust and efficient for the optimization needs. Furthermore, the idea of cascading was implemented in the present thesis in an ES context, with the Cascade Evolutionary Algorithm CEA(μ+λ), consisting of a number of ES(μ+λ) optimization stages. The main ES algorithm used, as well as the cascade ES scheme exhibited very good performance in the numerical applications where they were tested.

In an effort to improve the single objective optimization process and achieve faster convergence rates and better global behavior, other optimization algorithms were also studied and tested. One of them, namely the *Particle Swarm Optimization* (PSO) algorithm, a rather new methodology that has gained considerable attention during the last years in the academic community, was found to be very promising for use in structural optimum design. For the special needs of a constrained structural optimization problem, the traditional PSO algorithm was enhanced in the present thesis with a non-linear weight update rule and an efficient constraint handling technique that produced very good results in terms of the achieved optimum and the convergence history, compared to results from the literature. The proposed weight update rule showed to improve the efficiency of the optimization process, achieving better convergence behavior than other rules. The proposed constraint handling technique, based on a linear segment penalty function, showed very good performance, since it always led to feasible optimal designs, while also taking advantage of infeasible designs during the optimization procedure.

For further improving the computational efficiency of PSO for single-objective optimization problems, the PSO algorithm was also combined with a *Sequential Quadratic Programming* (SQP) mathematical programming method, in a new PSO-SQP algorithm. The hybrid algorithm proved to be a robust and efficient methodology for structural optimization problems, producing excellent results in the numerical applications where it was tested, achieving the same quality of results as the PSO methodology, in far better com-

putational time, by combining the global search abilities of the PSO with the local search abilities of the SQP algorithm.

9.1.2 Stochastic analysis

The stochastic analysis part of the methodology must be efficient, capable of yielding satisfactory results in demanding cases, where various stochastic variables are considered taking values from arbitrary probability density functions. In general, either analytical or sampling methodologies can be used, as described in detail in Chapter 2. Analytical methods, such as the *First- and Second-Order Reliability Methods* (FORM-SORM) and the *Response Surface Method* (RSM) exhibit significant limitations. FORM and SORM entail prior knowledge of the mean and the variance of each random variable, while a differentiable failure function is also required. Furthermore, they can be affected by the presence of various local optima for the optimization problem of finding the *Most Probable Point* (MPP). Also RSM requires a good approximation of the performance function around the design point and since the actual limit state function and the actual design point are not known a priori, the accuracy of the reliability estimate depends on the accuracy of the polynomial approximation.

In the present thesis, a *Monte Carlo Sampling* (MCS) methodology was used, having the strong advantage of being capable of handling every possible case, regardless of its complexity. The only drawback of the method is that it can be very computationally expensive, requiring in most of the cases an excessive number of simulations in order to provide reliable results. When combined with an optimization problem, the situation regarding the computational effort becomes even worse. For this reason, the MCS methodology was combined in the present thesis with special sampling methodologies, in particular *Importance Sampling* (IS) and *Latin Hypercube Sampling* (LHS). These methods proved to work very well by reducing significantly the computational cost of the purely random sampling methodology (Crude MCS).

The above enhancements for the stochastic analysis process were not implemented as a stand-alone procedure, but within a structural optimization algorithm, either with a single or with multiple objectives. The resultant RBDO, RDO and RRDO algorithms that were developed proved to be efficient for handling various test examples, ranging from simple plane structures to high-rise large-scale 3D structures.

9.1.3 Multi-objective optimization

In the context of optimum structural design considering uncertainties, a multi-objective optimization problem is encountered either in the RDO formulation or in RBDO formulations considering multiple criteria. In the present thesis, various methodologies were used and tested for solving the multi-objective optimization problem and providing the resultant *Pareto Front* (PF). The ES method combined with standard methods such as the *Linear Weighting Method* (LWM), the *Distance Function Method* (DFM) or the *Con-*

straint Method (CM), was first implemented. These schemes managed to give good results in some test examples.

Two new multi-objective methodologies were also proposed and tested, namely the ESMO algorithm, the CEATm cascade algorithm. A multi-objective structural optimization problem under dynamic loading was solved for the first time, with the *Direct Time Integration* (DTI) and the *Response Spectrum Modal Analysis* (RSMA) methods. The proposed algorithms proved to be efficient for providing a complete and detailed Pareto Front even in difficult cases where the multi-objective problem was not examined on its own, but within a RDO methodology.

9.1.4 Computational effort

Engineering problems that were once considered difficult or even impossible to handle due to their enormous computational requirements, can now be solved in a few seconds using a conventional personal computer. Nevertheless, despite the increased efficiency of high-performance today's computers, the available computing power is never enough to satisfy the demands of the researchers – engineers. New, more computationally demanding formulations of engineering problems arise, that need to be solved.

Compared to a simple linear analysis problem, an optimization problem can be considered three to four orders of magnitude more computationally demanding, or even more for complex structures with many design variables. Similarly, a stochastic analysis problem, where random variables are taken into account and the probability of structural failure or other statistical quantities have to be calculated, can be considered three to five orders of magnitude more computationally demanding than a linear analysis problem. This number can further increase if many random variables are taken into account and/or very small probabilities are to be calculated.

Thus, it is obvious that the combined problem of optimum structural design considering uncertainties is extremely computationally intensive. One of the objectives of the present thesis was to propose methodologies for improving the computational efficiency of the RBDO, RDO and RRDO algorithms that deal with the problem of structural optimization considering uncertainties. Although the computational cost was reduced, up to a certain point, using efficient optimizers and improved sampling techniques, it still required further reduction, especially for real-world large-scale problems with many design and/or random variables. In order to significantly reduce the computational effort and overcome the practical difficulties, the present thesis proposes the use of *Neural Network* (NN) metamodels to obtain estimates of the structural response required during the reliability analysis with the MCS method. The obtained estimates based on NN predictions have shown to be very satisfactory in terms of accuracy, while they lead to a reduction up to four orders of magnitude in computational time, making thus feasible the design optimization of realistic structures considering uncertainties.

9.2 Overall conclusions

Apart from the conclusions discussed in detail in Chapters 7 and 8 for every numerical application that was studied, the research work done for the thesis led to the following fundamental overall conclusions:

- Uncertainties in structural mechanics, analysis and design can play an extremely important role, affecting not only the safety and reliability of structures and mechanical components, but also the quality of their performance.
- The optimum design obtained by a deterministic optimization formulation can have limited value, as it can be severely affected by the uncertainties that are inherent in the model. The deterministic optimum can be associated with unaccepted probabilities of failure, or it can be vulnerable to slight variations of some uncertain parameters. The RBDO and RDO formulations can heal these problems, by taking into account the uncertainties in the optimum design process, as shown in the numerical applications sections of the thesis.
- The computational effort for considering the probabilistic optimum design is enormous. To reduce the computational burden, it is necesary to use efficient optimization algorithms and efficient sampling techniques for the stochastic analysis process.
- For further reducing the computational cost, NN metamodels can be applied as proposed in the thesis, providing acceptable numerical results at very low computational cost.
- The methodologies proposed in the thesis for the probabilistic optimum design can be applied to real-world structures. This is an indication that time is not far off when such design procedures will be common practice in structural engineering, possibly also adopted by the design codes.

The main objective of the thesis was to unify the concepts of probability-based safety analysis and structural optimization and provide the necessary numerical tools to deal with optimization problems considering uncertainties. The work aimed to uncover the advantages of the methodologies and to show how they can be applied in a practical way to deal with realistic structures.

The ultimate research goal is to establish the use of these methods as the state of the art in the near future and try to encourage practice towards that direction, away from the current use of safety factors and trial and error processes for the design of structural systems.

9.3 Future work

A research topic can be never considered as fully covered. It is clear that any piece of research work leaves many open issues for future research and sometimes it seems to raise

more questions than it has answered. Following the investigation described in this thesis, there are some natural extensions to this work that would help expand and strengthen the methodologies proposed and the obtained results:

- In an effort to further improve the computational efficiency of the methodologies described in the thesis, the extended implementation of parallel processing would be of great value. Parallel processing can be implemented in various parts of the methodologies:
  i. In the evolutionary optimization process, either with ES or with PSO, parallel processing can be implemented by assigning a processor to every member of the population, thus calculating the response of every generation or iteration at once.
  ii. Also for the SQP part of the hybrid PSO-SQP methodology, the gradients required for the line search with the global finite difference method can be calculated in parallel for every dimension of the problem, speeding up the SQP procedure.
  iii. In the multi-objective optimization process using scalarizing functions, parallel processing can be implemented by assigning a processor to every individual optimization run, thus calculating all the Pareto Front at once.
  iv. In a MCS procedure, either with LHS or in its basic form, parallel processing can be implemented by assigning a processor to every simulation, or to a number of simulations if the sample size is large, more than the number of the available processors. Then, the calculation of the statistical quantities can be done at once, or in a significantly lower number of runs.
- In order to further reduce the computational effort, different metamodels can be applied and tested, in comparison to the existing NN metamodels proposed in the present thesis. Considerable improvements in model-reduction methodologies in recent years have led to a growing interest in the field of *Reduced Order Models* (ROMs) across the simulation-based engineering science. These surrogate models can provide cost-efficient representations of large-scale computational systems and their implementation in a RBDO/RDO context would be of great interest (Weickum et al. 2008).
- The PSO methodology used in the thesis was applied to single objective optimization problems with continuous design variables, as the implementation of the method came at the end of the research work. It would be of interest to apply also a discrete version of PSO for structural optimization, where members can take values from given sets instead of continuous values, and test it in comparison to the discrete version of the ES methodology that was used in the thesis.
- The PSO can also be applied to multi-objective optimization problems, as an alternative to the ES-based methodologies that were used in this work. A comparison of the ES-based and PSO-based methods for multi-objective structural optimization

would be very interesting. The two methods share many similarities and consequently the same multi-objective methodologies can be possibly used, with minor modifications.

- The PSO algorithm was not tested as a part of an optimization procedure considering uncertainties. It would be very interesting to implement the PSO methodology in a RBDO/RDO context, as an alternative to the ES method used, highlight the differences in the methodologies and compare their results.

Bibliography

A parenthetical author–date referencing system has been used throughout the dissertation. Below is a list of the works cited, in alphabetical order.

Abido, M. A. (2007). "Two-level of nondominated solutions approach to multiobjective particle swarm optimization." *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, London, England.

Acar, E., and Haftka, R. T. (2007). "Reliability-based aircraft structural design pays, even with limited statistical data." *Journal of Aircraft*, 44(3), 812-823.

Adeli, H., and Kamal, O. (1986). "Efficient optimization of space trusses." *Comput. Struct.*, 24(3), 501-511.

Adeli, H., and Kamal, O. (1991). "Efficient optimization of plane trusses." *Advances in Engineering Software and Workstations*, 13(3), 116-122.

Adeli, H., and Park, H. S. (1995a). "Neural dynamics model for structural optimization - Theory." *Comput. Struct.*, 57(3), 383-390.

Adeli, H., and Park, H. S. (1995b). "Optimization of space structures by neural dynamics." *Neural Networks*, 8(5), 769-781.

Adeli, H., and Park, H. S. (1998). *Neurocomputing for Design Automation*, CRC Press, Inc., Boca Raton, FL, USA.

Agarwal, H., and Renaud, J. E. (2004). "Reliability based design optimization using response surfaces in application to multidisciplinary systems." *Engrg. Optim.*, 36(3), 291-311.

Anderson, E. C. (1999). "Monte Carlo Methods and Importance Sampling." in *Lecture notes for Stat 587C*.

Anderson, J., and Rosenfeld, E. (1989). *Neurocomputing*, MIT Press, Cambridge, MA.

Angeline, P. J. (1998). "Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences " in *Evolutionary Programming VII*, Springer Berlin / Heidelberg, 601-610.

Anthony, D. K., and Keane, A. J. (2003). "Robust-optimal design of a lightweight space structure using a genetic algorithm." *AIAA Journal*, 41(8), 1601-1604.

Arora, J. S. (1990). "Computational design optimization: A review and future directions." *Struct. Safety*, 7(2-4), 131-148.

Arslan, M. A., and Hajela, P. (1997). "Counterpropagation neural networks in decomposition based optimal design." *Comput. Struct.*, 65(5), 641-650.

Au, S.-K., and Beck, J. L. (2001). "Estimation of small failure probabilities in high dimensions by subset simulation." *Probabilist. Eng. Mech.*, 16(4), 263-277.

Bäck, T., and Schwefel, H.-P. (1993). "An Overview of Evolutionary Algorithms for Parameter Optimization." *Evol. Comput.*, 1(1), 1-23.

Barricelli, N. A. (1962). "Numerical testing of evolution theories." *ACTA Biotheoretica*, 16(3-4), 69-126.

Berke, L., Patnaik, S. N., and Murthy, P. L. N. (1993). "Optimum design of aerospace structural components using neural networks." *Comput. Struct.*, 48(6), 1001-1010.

Beyer, H.-G., and Sendhoff, B. (2007). "Robust optimization - A comprehensive survey." *Comput. Methods Appl. Mech. Engrg.*, 196(33-34), 3190-3218.

Bletzinger, K. U., Kimmich, S., and Ramm, E. (1991). "Efficient modeling in shape optimal design." *Comput. Syst. Eng.*, 2(5-6), 483-495.

Bochenek, B., and Foryś, P. (2006). "Structural optimization for post-buckling behavior using particle swarms." *Struct. Multidisc. Optim.*, 32(6), 521-531.

Breitung, K. (1984). "Asymptotic Approximations for Multinormal Integrals." *J. Engrg. Mech.*, 110(3), 357-366.

Bremicker, M., Papalambros, P. Y., and Loh, H. T. (1990). "Solution of mixed-discrete structural optimization problems with a new sequential linearization algorithm." *Comput. Struct.*, 37(4), 451-461.

Buchanan, J. T. (1986). "Multiple objective mathematical programming: A review." *New Zealand Operational Research*, 14(1), 1-27.

Bucher, C. G. (1988). "Adaptive sampling - an iterative fast Monte Carlo procedure." *Struct. Safety*, 5, 119-126.

Cai, J. (1995). "Discrete Optimization of Structures under Dynamic Loading by Using Sequential and Parallel Evolution Strategies." PhD Thesis, University of Essen, Germany.

Carmichael, D. G. (1981). *Structural modelling and optimization*, Ellis Horwood Ltd (John Wiley and Sons Ltd), Chichester, England.

CEN. (1991). "EN 1991-1: Eurocode 1, Actions on structures, Part 1." European Committee for Standardization (CEN).

CEN. (1993). "EN 1993-1: Eurocode 3, Design of steel structures, Part 1." European Committee for Standardization (CEN).

CEN. (1998). "EN 1998-1: Eurocode 8, Design of structures for earthquake resistance, Part 1." European Committee for Standardization (CEN).

Chankong, V., and Haimes, Y. Y. (1983). *Multiobjective Decision Making Theory and Methodology*, Elsevier Science, New York.

Charmpis, D. C., Lagaros, N. D., and Papadrakakis, M. (2005). "Multi-database exploration of large design spaces in the framework of cascade evolutionary structural sizing optimization." *Comput. Methods Appl. Mech. Engrg.*, 194(30-33), 3315-3330.

Chateauneuf, A. (2008). "Principles of reliability-based design optimization." in *Structural design optimization considering uncertainties*, Y. Tsompanakis, N. D. Lagaros, and M. Papadrakakis, eds., Taylor & Francis, 3-30.

Chen, S., Mei, T., Luo, M., and Yang, X. (2007). "Identification of nonlinear system based on a new hybrid gradient-based PSO algorithm." *Proceedings of the 2007 International Conference on Information Acquisition, ICIA*, 265-268.

Chen, W., and Lewis, K. (1999). "A robust design approach for achieving flexibility in multidisciplinary design." *AIAA Journal*, 37(8), 982-990.

Coelho, L. D. S., and Mariani, V. C. (2007). "Particle swarm optimization with quasi-newton local search for solving economic dispatch problem." *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 3109-3113.

Coelho, R. F., Bersini, H., and Bouillard, P. (2003). "Parametrical mechanical design with constraints and preferences: application to a purge valve." *Comput. Methods Appl. Mech. Engrg.*, 192(39-40).

Coello Coello, C. A. (2000). "An updated survey of GA-based multi-objective optimization techniques." *ACM Computing Surveys*, 32(2), 109-143.

Coello Coello, C. A., and Lechuga, M. S. (2002). "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization." *Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence*, Hawaii, May 12-17, 1051-1056.

Cohon, J. L. (1978). *Multi-objective Programming and Planning*, Academic Press, New York.

Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust-Region Methods*, SIAM, Philadelphia, USA.

Dariani, S., Keshavarz, M., Parviz, M., Raoufy, M. R., and Gharibzadeh, S. (2007). "Modeling force-velocity relation in skeletal muscle isotonic contraction using an artificial neural network." *BioSystems*, 90(2), 529-534.

Darwin, C. (1859). *On the Origin of Species*, London.

Das, I., and Dennis, J. E. (1998). "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems." *SIAM J. on Optimization*, 8(3), 631-657.

Das, S., Koduru, P., Gui, M., Cochran, M., Wareing, A., Welch, S. M., and Babin, B. R. (2006). "Adding local search to particle swarm optimization." *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, 428-433.

Deb, K. (1999). "Evolutionary algorithms for multi-criterion optimization in engineering design." in *Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN '99)*, K. Miettinen, M. Mäkelä, P. Neittaanmäki, and J. Périaux, eds., 135-161.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE Trans. Evol. Comput.*, 6(2), 182-197.

Demuth, H., Beale, M., and Hagan, M. (2008). *Matlab Neural Network Toolbox 6 User's Guide*, The MathWorks, Inc.

Der Kiureghian, A., and Dakessian, T. (1998). "Multiple design points in first and second-order reliability." *Struct. Safety*, 20(13), 37-49.

Der Kiureghian, A., and Ditlevsen, O. (2009). "Aleatory or epistemic? Does it matter?" *Struct. Safety*, 31(2), 105-112.

Der Kiureghian, A., Lin, H.-Z., and Hwang, S.-J. (1987). "Second-Order Reliability Approximations." *J. Engrg. Mech.*, 113(8), 1208-1225.

Dimopoulos, G. G. (2007). "Mixed-variable engineering optimization based on evolutionary and social metaphors." *Comput. Methods Appl. Mech. Engrg.*, 196, 803-817.

Diwekar, U. (2008). *Introduction to Applied Optimization*, Springer.

Dobbs, M. W., and Nelson, R. B. (1976). "Application of optimality criteria to automated structural design." *AIAA Journal*, 14(10), 1436-1443.

Doltsinis, I., and Kang, Z. (2004). "Robust design of structures using optimization methods." *Comput. Methods Appl. Mech. Engrg.*, 193, 2221-2237.

Doltsinis, I., Kang, Z., and Cheng, G. (2005). "Robust design of non-linear structures using optimization methods." *Comput. Methods Appl. Mech. Engrg.*, 194(12-16), 1779-1795.

Du, L., Choi, K. K., Youn, B. D., and Gorsich, D. (2006). "Possibility-based design optimization method for design problems with both statistical and fuzzy input data." *J. Mech. Des. - T. ASME*, 128(4), 928-935.

El-Sayed, M. E. M., and Jang, T. S. (1994). "Structural optimization using unconstrained nonlinear goal programming algorithm." *Comput. Struct.*, 52(4), 723-727.

Eschenauer, H. A., Schumacher, A., and Vietor, T. (1993). "Decision makings for initial designs made of advanced materials." *Topology Design of Structures: Proceedings of the NATO Advanced Workshop (NATO Science Series: E)* Sesimbra, Portugal, 469-480.

Fausett, L. V. (1994). *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice-Hall, Inc.

Fieldsend, J. E., and Singh, S. (2002). "A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence." *Proceedings of UK Workshop on Computational Intelligence (UKCI '02)*, Birmingham, UK, September 2-4, 34-44.

Fiessler, B., Rackwitz, R., and Neumann, H.-J. (1979). "Quadratic Limit States in Structural Reliability." *Journal of the Engineering Mechanics Division ASCE*, 105(4), 661-676.

Fishman, G. S., and Huang, B. D. (1983). "Antithetic variates revisited." *Communications of the ACM*, 26(11), 964-971.

Fleury, C. (1993). "Dual methods for convex separable problems." *Optimization of large structural systems: Proceedings of the NATO/DFG Advanced Study Institute*, Berchtesgaden, Germany, September 23-October 4, 509-530.

Florian, A. (1992). "An efficient sampling scheme: Updated Latin Hypercube Sampling." *Probabilist. Eng. Mech.*, 7(2), 123-130.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*, John Wiley & Sons, New York.

Fonseca, C. M., and Fleming, P. J. (1993). "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization." *Proceedings of the 5th International Conference on Genetic Algorithms*, San Mateo, California, 416-423.

Fonseca, C. M., and Fleming, P. J. (1995). "An Overview of Evolutionary Algorithms in Multi-objective Optimization." *Evol. Comput.*, 3(1), 1-16.

Fourie, P., and Groenwold, A. (2001). "The particle swarm optimization in topology optimization." *Fourth world congress of structural and multidisciplinary optimization*, Dalian, China.

Fourie, P. C., and Groenwold, A. (2002). "The particle swarm optimization algorithm in size and shape optimization." *Struct. Multidisc. Optim.*, 23(4), 259–267.

Frangopol, D. M. (1995). "Reliability-Based Optimum Structural Design)." in *Probabilistic Structural Mechanics Handbook*, C. Sundararajan, ed., Chapman & Hall, New York, 352-387.

Frangopol, D. M., and Maute, K. (2003). "Life-cycle reliability-based optimization of civil and aerospace structures." *Comput. Struct.*, 81(7), 397-410.

Galante, M. (1992). "Structures optimization by a simple genetic algorithm." *Numerical Methods In Engineering And Applied Sciences*, Barcelona, Spain, 862-870.

Gasser, M., and Schuëller, G. I. (1997). "Reliability-Based Optimization of structural systems." *Math. Meth. Oper. Res.*, 46(3), 287-307.

Gellatly, R. A., and Berke, L. (1971). *Optimal structural design*, AFFDL-TR-70-165, Air Force Flight Dynamics Lab, Wright-Patterson Air Force Base, Ohio.

Ghasemi, M. R., Hinton, E., and Wood, R. D. (1997). "Optimization of trusses using genetic algorithms for discrete and continuous variables." *Eng. Computations*, 16 (3), 272-303.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1986). *User's guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming, Technical Report SOL 86-2*, Stanford University, Dept. of Operations Research.

Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical Optimization*, Academic Press.

Gisakis, A., Tsirigas, P., Plevris, V., and Papadrakakis, M. (2009). "Evaluation Study of the Improved TRIC Shell Element for Linear and Nonlinear Structural Analysis." *2nd South-East European Conference on Computational Mechanics (SEECCM 2009)*, Rhodes, Greece, June 22-24.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley Longman Publishing Co., Boston, Massachusetts.

Goldberg, D. E., and Deb, K. (1991). "A comparative analysis of selection schemes used in genetic algorithms." in *Foundations of Genetic Algorithms*, Morgan Kaufmann.

Gray, W. A., and Melchers, R. E. (2006). "Modifications to the 'directional simulation in the load space' approach to structural reliability analysis " *Probabilist. Eng. Mech.*, 21(2), 148-158.

Greiner, D., Emperador, J. M., and Winter, G. (2004). "Single and multiobjective frame optimization by evolutionary algorithms and the auto-adaptive rebirth operator." *Comput. Methods Appl. Mech. Engrg.*, 193(33-35), 3711-3743.

Gunaratnam, D. J., and Gero, J. S. (1994). "Effect of representation on the performance of neural networks in structural engineering applications." *Microcomputers in Civil Engineering*, 9(2), 97-108.

Gunawan, S., and Papalambros, P. Y. (2006). "A bayesian approach to reliability-based optimization with incomplete information." *J. Mech. Des. - T. ASME*, 128(4), 909-918.

Haftka, R. T., and Gürdal, Z. (1992). *Elements of Structural Optimization*, Kluwer Academic Publishers.

Hagan, M. T., Demuth, H. B., and Beale, M. H. (1996). *Neural Network Design*, PWS Publishing, Boston, MA.

Hager, K., and Balling, R. (1988). "New approach for discrete structural optimization." *J. Struct. Eng. (ASCE)*, 114(5), 1120 - 1133.

Haimes, Y. Y., and Hall, W. A. (1974). "Multiobjectives in water resource systems analysis: the surrogate worth trade off method." *Water Resour. Res.*, 10(4), 615-623.

Hajela, P., and Berke, L. (1991). "Neurobiological computational models in structural analysis and design." *Comput. Struct.*, 41, 657-667.

Hasofer, A. M., and Lind, N. C. (1974). "Exact and invariant second moment code format." *J. Engrg. Mech.*, 100(1), 111-121.

Hassan, R., Cohanim, B., de Weck, O., and Venter, G. (2005). "A Comparison of Particle Swarm Optimization and the Genetic Algorithm." *Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, Austin, Texas, USA, April 18-21.

Haug, E. J., and Arora, J. S. (1979). *Applied Optimal Design: Mechanical and Structural Systems*, John Wiley & Sons Inc, New York.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*, Prentice Hall.

He, Q., and Wang, L. (2007). "An effective co-evolutionary particle swarm optimization for constrained engineering design problems." *Eng. Appl. Artif. Intell.*, 20(1), 89-99.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*, New York: Wiley.

Hemelrijk, J. (1966). "Underlining random variables." *Statistica Neerlandica*, 20(1).

Hinton, E., and Sienz, J. (1993). "Fully stressed topological design of structures using an evolutionary procedure." *Eng. Computations*, 12, 229-244.

Hinton, E., and Sienz, J. (1994). "Aspects of adaptive finite element analysis and structural optimization." in *Advances in Structural Optimization*, B. H. V. Topping and M. Papadrakakis, eds., Civil-Comp Press, Edinburgh, 1-26.

Hoffmeister, F., and Back, T. (1991). "Genetic Algorithms and Evolution Strategies-Similarities and Differences." in *Parallel Problems Solving from Nature*, H. P. Schwefel and R. Manner, eds., Springel-Verlag, Berlin, Germany, 455-469.

Hohenbichler, M., and Rackwitz, R. (1988). "Improvement of Second-Order Reliability Estimates by Importance Sampling." *J. Engrg. Mech.*, 114(12), 2195-2199.

Holland, J. (1975). *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor.

Hong, H. P. (1999). "Simple Approximations for Improving Second-Order Reliability Estimates." *J. Engrg. Mech.*, 125(5), 592-595.

Hooke, R., and Jeeves, T. A. (1961). "Direct Search Solution of Numerical and Statistical Problems." *Journal of the ACM*, 8(2), 212-229.

Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences*, 79(8), 2554-2558.

Horn, J. (1997). "Multicriterion decision making." in *Handbook of Evolutionary Computation*, T. Back, D. B. Fogel, and Z. Michalewicz, eds., Oxford Univ. Press, F1.9:1-F1.9:15.

Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). "A niched pareto genetic algorithm for multiobjective optimization." *Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Orlando, Florida, USA, June 27-29, 82-87.

Hornik, K. (1991). "Approximation capabilities of multilayer feedforward networks." *Neural Networks*, 4(2), 251-257.

Hu, X., Eberhart, R., and Shi, Y. (2003). "Engineering Optimization with Particle Swarm." *IEEE Swarm Intelligence Symposium (SIS 2003)*, Indianapolis, Indiana, USA, 53-57.

Hu, X., and Eberhart, R. C. (2002). "Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization." *Proceedings of the 2002 Congess on Evolutionary Computation, part of the 2002 IEEE World Congress on Computational Intelligence*, Hawaii, May 12-17, 2002.

Hurtado, J. E. (2008). "Structural robustness and its relationship to reliability." in *Structural design optimization considering uncertainties*, Y. Tsompanakis, N. D. Lagaros, and M. Papadrakakis, eds., Taylor & Francis, 435-470.

Hurtado, J. E., and Alvarez, D. A. (2001). "Neural-network-based reliability analysis: A comparative study." *Comput. Methods Appl. Mech. Engrg.*, 191(1-2), 113-132.

Hwang, C. L., and Masud, A. S. M. (1979). *Multiple objective decision making - Methods and applications: a state-of-the-art survey*, Springer-Verlag, Berlin.

ISO. (1993). "ISO Standards Handbook. Quantities and Units." International Organization for Standardization.

Izui, K., Nishiwaki, S., and Yoshimura, M. (2005). "Swarm Optimization Algorithms Incorporating Design Sensitivities." *6th World Congress on Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil.

Izui, K., Nishiwaki, S., and Yoshimura, M. (2007). "Swarm algorithms for single - and multi - objective optimization problems incorporating sensitivity analysis." *Engrg. Optim.*, 39(8).

James, B. A. P. (1985). "Variance Reduction Techniques." *Journal of the Operational Research Society*, 36, 525-530.

Jensen, H. A., Valdebenito, M. A., and Schueller, G. I. (2008). "An efficient reliability-based optimization scheme for uncertain linear systems subject to general gaussian excitation." *Comput. Methods Appl. Mech. Engrg.*, 198(1), 72-87.

Jiang, C., Han, X., and Liu, G. R. (2007). "Optimization of structures with uncertain constraints based on convex model and satisfaction degree of interval." *Comput. Methods Appl. Mech. Engrg.*, 196(49-52), 4791-4800.

Jiang, M., Corotis, R. B., and Hugh Ellis, J. (2000). "Optimal life-cycle costing with partial observability." *J. Infrastruct. Syst.*, 6 (2), 56-66.

Jung, D. H., and Lee, B. C. (2002). "Development of a simple and efficient method for robust optimization." *Int. J. Numer. Meth. Engng.*, 53, 2201-2215.

Kalagnanam, J. R., and Diwekar, U. M. (1997). "An efficient sampling technique for off-line quality control." *Technometrics*, 39(3), 308-319.

Kaveh, A., and Talatahari, S. (2008). "A hybrid particle swarm and ant colony optimization for design of truss structures." *Asian Journal of Civil Engineering (building and housing)*, 9(4), 329-348.

Kennedy, J., and Eberhart, R. (1995). "Particle swarm optimization." *IEEE International Conference on Neural Networks*, Piscataway, NJ, USA, 1942–1948.

Kennedy, J., and Mendes, R. (2002). "Population structure and particle swarm performance." *Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02)*, 1671-1676.

Khan, A. I., Topping, B. H. V., and Bahreininejad, A. (1993). "Parallel training of neural networks for finite element mesh generation." in *Neural Networks & Combinatorial Optimization in Civil & Structural Engineering*, B. H. V. Topping and A. I. Khan, eds., Civil-Comp Press, Edinburgh, 81-94.

Kim, D. (1999). "Normalization methods for input and output vectors in backpropagation neural networks." *International Journal of Computer Mathematics*, 71(2), 161-171.

Koch, P. N., Yang, R. J., and Gu, L. (2004). "Design for six sigma through robust optimization." *Struct. Multidisc. Optim.*, 26(3-4), 235-248.

Koski, J. (1984). "Bicriterion optimum design method for elastic trusses." *Acta Polytechnica Scandinavica, Mechanical engineering series*, 86.

Koski, J. (1994). "Multicriterion structural optimization." in *Advances in Design Optimization*, H. Adeli, ed., Chapman & Hall, London, UK, 194-224.

Koutsourelakis, P. S., Pradlwarter, H. J., and Schuëller, G. I. (2004). "Reliability of structures in high dimensions, part I: Algorithms and applications." *Probabilist. Eng. Mech.*, 19(4), 409-417.

Köylüoglu, H. U., and Nielsen, S. R. K. (1994). "New approximations for SORM integrals." *Struct. Safety*, 13(4), 235-246.

Koziel, S., and Michalewicz, Z. (1999). "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization." *Evol. Comput.*, 7(1), 19-44.

Krose, B., and van der Smagt, P. (1996). *An introduction to Neural Networks*, The University of Amsterdam.

Kuschel, N., and Rackwitz, R. (1997). "Two basic problems in reliability-based structural optimization." *Math. Meth. Oper. Res.*, 46, 309-333.

Kuschel, N., and Rackwitz, R. (2000). "Optimal design under time-variant reliability constraints." *Struct. Safety*, 22(2), 113-127.

Kuźniar, K., and Waszczyzyn, Z. (2007). "Neural Networks for the Simulation and Identification Analysis of Buildings Subjected to Paraseismic Excitations." in *Intelligent Computational Paradigms in Earthquake Engineering*, N. D. Lagaros and Y. Tsompanakis, eds., IGI Publishing.

L'Ecuyer, P., and Buist, E. (2008). "On the interaction between stratification and control variates, with illustrations in a call centre simulation." *Journal of Simulation*, 2(1), 29-40.

Lagaros, N. D., Charmpis, D. C., and Papadrakakis, M. (2005a). "An adaptive neural network strategy for improving the computational performance of evolutionary structural optimization." *Comput. Methods Appl. Mech. Engrg.*, 194(30-33), 3374-3393.

Lagaros, N. D., Fragiadakis, M., and Papadrakakis, M. (2004). "Optimum design of shell structures with stiffening beams." *AIAA Journal*, 42(1), 175-184.

Lagaros, N. D., Garavelas, A. T., and Papadrakakis, M. (2008a). "Innovative seismic design optimization with reliability constraints." *Comput. Methods Appl. Mech. Engrg.*, 198(1), 28-41.

Lagaros, N. D., and Papadrakakis, M. (2007). "Robust seismic design optimization of steel structures." *Struct. Multidisc. Optim.*, 33(6), 457-469.

Lagaros, N. D., Papadrakakis, M., and Kokossalakis, G. (2002). "Structural optimization using evolutionary algorithms." *Comput. Struct.*, 80(7-8), 571-589.

Lagaros, N. D., Papadrakakis, M., and Plevris, V. (2005b). "Multi-objective Optimization of Space Structures under Static and Seismic Loading Conditions." in *Evolutionary Multiobjective Optimization*, A. Abraham, L. C. Jain, and R. Goldberg, eds., Springer, 273-300.

Lagaros, N. D., Plevris, V., and Papadrakakis, M. (2005c). "Multi-objective Design Optimization Using Cascade Evolutionary Computations." *Comput. Methods Appl. Mech. Engrg.*, 194(30-33), 3496-3515.

Lagaros, N. D., Plevris, V., and Papadrakakis, M. (2007). "Reliability Based Robust Design Optimization of Steel Structures." *Int. J. Simul. Multidisci. Des. Optim.*, 1(1), 19-29.

Lagaros, N. D., Plevris, V., and Papadrakakis, M. (2009). "Neurocomputing Strategies for Solving Reliability-Robust Design Optimization Problems." *Eng. Computations*, Submitted for publication.

Lagaros, N. D., Plevris, V., Tsompanakis, Y., and Papadrakakis, M. (2005d). "Multi-performance Robust Optimum Design of Steel Structures." *6th World Congress*

on Structural and Multidisciplinary Optimization (WCSMO 6), Rio de Janeiro, Brazil, May 30 - June 3.

Lagaros, N. D., Tsompanakis, Y., Fragiadakis, M., Plevris, V., and Papadrakakis, M. (2008b). "Metamodel-based Computational Techniques for Solving Structural Optimization Problems Considering Uncertainties." in *Structural Design Optimization Considering Uncertainties*, Y. Tsompanakis, N. D. Lagaros, and M. Papadrakakis, eds., Taylor and Francis, 567-597.

Lamberti, L., and Pappalettere, C. (2004). "Improved sequential linear programming formulation for structural weight minimization." *Comput. Methods Appl. Mech. Engrg.*, 193(33-35), 3493-3521.

Lasdon, L. S., Warren, A. D., Jain, A., and Ratner, M. (1978). "Design and testing of a generalized reduced gradient code for nonlinear programming." *ACM Transactions on Mathematical Software*, 4(1), 34-50.

Lee, K. H., and Park, G. J. (2001). "Robust optimization considering tolerances of design variables." *Comput. Struct.*, 79, 77-86.

Leonard, J. A., Kramer, M. A., and Ungar, L. H. (1992). "A neural network architecture that computes its own reliability." *Comput. Chem. Eng.*, 16(9), 819-835.

Lettvin, J. Y., Maturana, H. R., McCulloch, W. S., and Pitts, W. H. (1959). "What the Frog's Eye Tells the Frog's Brain." *Proceedings of the Institute of Radic Engineers*, 47(11), 1940-1951.

Li, L. J., Huang, Z. B., Liu, F., and Wu, Q. H. (2007). "A heuristic particle swarm optimizer for optimization of pin connected structures." *Comput. Struct.*, 85, 340-349.

Liang, J., Mourelatos, Z. P., and Nikolaidis, E. (2007). "A single-loop approach for system reliability-based design optimization." *J. Mech. Des. - T. ASME*, 129(12), 1215-1224.

Liang, J. J., and Suganthan, P. N. (2006). "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism." *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 9-16.

Lieberman, E. R. (1991). "Soviet Multi-Objective Mathematical Programming Methods: An Overview." *Management Science*, 37(9), 1147-1165.

Lohninger, H. (1993). "Evaluation of neural networks based on radial basis functions and their application to the prediction of boiling points from structural parameters." *Journal of Chemical Information and Computer Sciences*, 33(5), 736-744.

Luh, G. C., and Chueh, C. H. (2004). "Multi-modal topological optimization of structure using immune algorithm." *Comput. Methods Appl. Mech. Engrg.*, 193(36-38), 4035-4055.

Maa, C., and Shanblatt, M. A. (1992). "A two-phase optimization neural network." *IEEE Trans. Neural Networks*, 3(6), 1003-1009.

Makris, P. A., Provatidis, C. G., and Rellakis, D. A. (2006). "Discrete variable optimization of frames using a strain energy criterion." *Struct. Multidisc. Optim.*, 31(5), 410-417.

Marler, R. T., and Arora, J. S. (2004). "Survey of multi-objective optimization methods for engineering." *Struct. Multidisc. Optim.*, 26(6), 369-395.

McAllister, C. D., and Simpson, T. W. (2003). "Multidisciplinary robust design optimization of an internal combustion engine." *J. Mech. Des. - T. ASME*, 125(1), 124-130.

McCorkle, D. S., Bryden, K. M., and Carmichael, C. G. (2003). "A new methodology for evolutionary optimization of energy systems." *Comput. Methods Appl. Mech. Engrg.*, 192(44-46), 5021-5036.

McCulloch, W. S., and Pitts, W. (1943). "A logical calculus of the ideas immanent in nervous activity." *Bulletin of Mathematical Biophysics*, 5, 115-133.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code." *Technometrics*, 21(2), 239-245.

Melchers, R. E. (1989). "Importance sampling in structural systems." *Struct. Safety*, 6, 3-10.

Memari, A. M., and Fuladgar, A. (1994). "Minimum Weight Design of Trusses by BEHSAZ Program." *Proceedings of 2nd International Conference on Computational Structures Technology*, Athens, Greece, August 30 - September 1.

Mendes, R., Kennedy, J., and Neves, J. (2004). "The fully informed particle swarm: simpler, maybe better." *IEEE Trans. Evol. Comput.*, 8(3), 204-210.

Messac, A., and Ismail-Yahaya, A. (2002). "Multi-objective robust design using physical programming." *Struct. Multidisc. Optim.*, 23(5), 357-371.

Metropolis, N. (1987). "The Beginning of Monte Carlo Method." *Los Alamos Science*(Special Issue dedicated to Stanislaw Ulam), 125-130.

Metropolis, N., and Ulam, S. (1949). "The Monte Carlo method." *Journal of the American Statistical Association*, 44(247), 335-341.

Mezura-Montes, E., and Lopez-Ramirez, B. C. (2007). "Comparing bio-inspired algorithms in constrained optimization problems." *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 662-669.

Michalewicz, Z. (1995). "A Survey of Constraint Handling Techniques in Evolutionary Computation Methods." *Proceedings of the 4th Annual Conference on Evolutionary Programming* San Diego, California, 135-155.

Miettinen, K. (1999). *Nonlinear multiobjective optimization*, Kluwer Academic Publishers, Boston.

Minsky, M., and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*, The MIT Press.

Mori, Y., and Ellingwood, B. R. (1993). "Time-dependent system reliability analysis by adaptive importance sampling." *Struct. Safety*, 12(1), 59-73.

Moses, F. (1997). "Problems and prospects of reliability-based optimization." *Eng. Struct.*, 19(4), 293-301.

Munoz-Zavala, A. E., Hernandez-Aguirre, A., Villa-Diharce, E. R., and Botello-Rionda, S. (2006). "PESO+ for Constrained Optimization." *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 231-238.

Myung, H., Kim, J.-H., and Fogel, D. B. (1995). "Preliminary investigation into a two-stage method of evolutionary optimization on constrained problems." *Proceedings of the 4th Annual Conference on Evolutionary Programming*, San Diego, California, 449-463.

Nie, J., and Ellingwood, B. R. (2004). "A new directional simulation method for system reliability. Part II: Application of neural networks." *Probabilist. Eng. Mech.*, 19(4), 437-447.

Nie, J., and Ellingwood, B. R. (2005). "Finite Element-Based Structural Reliability Assessment Using Efficient Directional Simulation." *J. Engrg. Mech.*, 131(3), 259-267.

Nikolaidis, E. (2007). "Decision-based approach for reliability design." *J. Mech. Des. - T. ASME*, 129(5), 466-475.

Nocedal, J., and Wright, S. J. (1999). *Numerical Optimization*, Springer, New York.

Nowak, A. S., and Collins, K. R. (2000). *Reliability of structures*, McGraw-Hill.

Olhoff, N., Rasmussen, J., and Lund, E. (1992). "Method of exact numerical differentiation for error estimation in finite element based semi-analytical shape sensitivity analyses." Institute of mechanical Engineering, Aalborg University, Special Report No. 10, Aalborg, DK.

Omkar, S. N., Mudigere, D., Narayana Naik, G., and Gopalakrishnan, S. (2008). "Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures." *Comput. Struct.*, 86(1-2), 1-14.

Owen, A. B. (1997). "Monte carlo variance of scrambled net quadrature." *SIAM Journal on Numerical Analysis*, 34(5), 1884-1910.

Papadrakakis, M., and Lagaros, N. D. (2000). "Advances in structural optimization." in *Recent Advances in Mechanics, Honorary Volume for Professor A.N. Kounadis*, J. T. Katsikadelis, D. E. Beskos, and E. E. Gdoutos, eds., NTUA Publics.

Papadrakakis, M., and Lagaros, N. D. (2002). "Reliability-based structural optimization using neural networks and Monte Carlo simulation." *Comput. Methods Appl. Mech. Engrg.*, 191(32), 3491-3507.

Papadrakakis, M., Lagaros, N. D., and Fragakis, Y. (2003). "Parallel computational strategies for structural optimization." *Int. J. Numer. Meth. Engng.*, 58(9), 1347-1380.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2000a). "Optimum Design of 3D Structures under Static and Dynamic Loading Conditions." *Fourth International Colloquium on Computation of Shell & Spatial Structures (IASS-IACM 2000)*, Chania, Greece, June 4-7.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2000b). "Optimum Design of Structures under Seismic Loading." *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2000)*, Barcelona, Spain, September 11-14.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2001a). "Optimum Design of Space Frames under Seismic Loading." *Int. J. Struct. Stabil. Dyn.*, 1(1), 105-123.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2002a). "Multi-objective Optimization of Skeletal Structures under Static and Seismic Loading Conditions." *Engrg. Optim.*, 34(6), 645-669.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2002b). "Multi-objective Optimum Design of 3D Structures under Static and Seismic Loading Conditions." *4th GRACM Congress on Computational Mechanics*, Patra, Greece, June 27-29.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2004a). "Structural Optimization Considering the Probabilistic System Response." *Theoret. Appl. Mech.*, 31(3-4), 361-394.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2004b). "Structural Optimization Considering the Probabilistic System Response." *The First International Conference on Computational Mechanics (CM'04)*, Belgrade, Serbia, November 15-17.

Papadrakakis, M., Lagaros, N. D., and Plevris, V. (2005). "Design optimization of steel structures considering uncertainties." *Eng. Struct.*, 27(9), 1408-1418.

Papadrakakis, M., Lagaros, N. D., Plevris, V., Charmpis, D. C., and Fragiadakis, M. (2006). "Computational Challenges in Optimum Structural Design." *7th World Congress on Computational Mechanics (WCCM VII)*, Los Angeles, California, USA, July 16-22.

Papadrakakis, M., Lagaros, N. D., Thierauf, G., and Cai, J. (1998a). "Advanced solution methods in structural optimization based on evolution strategies." *Eng. Computations*, 15(1), 12-34.

Papadrakakis, M., Lagaros, N. D., and Tsompanakis, Y. (1998b). "Structural optimization using evolution strategies and neural networks." *Comput. Methods Appl. Mech. Engrg.*, 156(1-4), 309-333.

Papadrakakis, M., Lagaros, N. D., Tsompanakis, Y., and Plevris, V. (2001b). "Large Scale Structural Optimization: Computational Methods and Optimization Algorithms." *Archives of Computational Methods in Engineering*, 8(3), 239-301.

Papadrakakis, M., and Papadopoulos, V. (1995). "A computationally efficient method for the limit elasto plastic analysis of space frames." *Comput. Mech.*, 16(2), 132-141.

Papadrakakis, M., Papadopoulos, V., and Lagaros, N. D. (1996a). "Structural reliability analyis of elastic-plastic structures using neural networks and Monte Carlo simulation." *Comput. Methods Appl. Mech. Engrg.*, 136(1-2), 145-163.

Papadrakakis, M., Plevris, V., Lagaros, N. D., and Papadopoulos, V. (2004c). "Robust Design Optimization of 3D Truss Structures Using Evolutionary Computation." *6th World Congress on Computational Mechanics (WCCM VI) in conjunction with APCOM'04*, Beijing, China, September 5-10.

Papadrakakis, M., and Tsompanakis, Y. (1999). "Domain decomposition methods for parallel solution of shape sensitivity analysis problems." *Int. J. Numer. Meth. Engng.*, 44(2), 281-303.

Papadrakakis, M., Tsompanakis, Y., Hinton, E., and Sienz, J. (1996b). "Advanced solution methods in topology optimization and shape sensitivity analysis." *Eng. Computations*, 13(5), 57-90.

Papadrakakis, M., Tsompanakis, Y., and Lagaros, N. D. (1999). "Structural Shape Optimization using Evolution Strategies." *Engrg. Optim.*, 31(4), 515-540.

Pareto, V. (1896). *Cours d' economique politique*, Rouge, Lausanne.

Park, G. J., Lee, T. H., Lee, K. H., and Hwang, K. H. (2006). "Robust design: An overview." *AIAA Journal*, 44 (1), 181-191.

Parkinson, M. B., Reed, M. P., Kokkolaras, M., and Papalambros, P. Y. (2007). "Optimizing truck cab layout for driver accommodation." *J. Mech. Des. - T. ASME*, 129(11), 1110-1117.

Parsopoulos, K. E., Tasoulis, D. K., and Vrahatis, M. N. (2004). "Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization." *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*.

Parsopoulos, K. E., and Vrahatis, M. N. (2002). "Particle swarm optimization method in multiobjective problems." *Proceedings of the 2002 ACM symposium on Applied computing*, Madrid, Spain, 603-607.

Parzen, E. (1962). "On Estimation of a Probability Density Function and Mode." *Ann. Math. Statist.*, 33(3), 1065-1076.

Patnaik, S. N., Coroneos, R. M., Guptill, J. D., and Hopkins, D. A. (1996). "Comparative evaluation of different optimization algorithms for structural design applications." *Int. J. Numer. Meth. Engng.*, 39(10), 1761-1774.

Patnaik, S. N., Coroneos, R. M., and Hopkins, D. A. (1997). "A cascade optimization strategy for solution of difficult design problems." *Int. J. Numer. Meth. Engng.*, 40(12), 2257 - 2266.

Pedersen, P. (1993). "Topology optimization of three dimensional trusses." *Topology Design of Structures: Proceedings of the NATO Advanced Workshop (NATO Science Series: E)* Sesimbra, Portugal, 19-31.

Perez, R. E., and Behdinan, K. (2007a). "Particle swarm approach for structural design optimization." *Comput. Struct.*, 85, 1579-1588.

Perez, R. E., and Behdinan, K. (2007b). "Particle Swarm Optimization in Structural Design." in *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, F. T. S. Chan and M. K. Tiwari, eds., Itech Education and Publishing, Vienna, Austria, 373-394.

Plevris, V., Lagaros, N. D., Charmpis, D. C., and Papadrakakis, M. (2006a). "Metamodel Assisted Techniques for Structural Optimization." *First South-East European Conference on Computational Mechanics (SEECCM 06)*, Kragujevac, Serbia, June 28-30.

Plevris, V., Lagaros, N. D., and Papadrakakis, M. (2005a). "Robust Design Optimization of Steel Structures." *5th GRACM International Congress on Computational Mechanics*, Limassol, Cyprus, June 29 - July 1.

Plevris, V., Lagaros, N. D., and Papadrakakis, M. (2005b). "Robust Design Optimization of Steel Structures Using Cascade Evolutionary Computations." *Eighth U.S. National Congress on Computational Mechanics (USNCCM 8)*, Austin Texas, USA, July 24-28.

Plevris, V., Lagaros, N. D., and Papadrakakis, M. (2006b). "Advances in Evolutionary Structural Optimization Considering Uncertainties." *7th World Congress on Computational Mechanics (WCCM VII)*, Los Angeles, California, USA, July 16-22.

Plevris, V., Lagaros, N. D., and Papadrakakis, M. (2007). "Seismic Design Optimization of Steel Structures Considering Uncertainties." *Computational Methods in Structural Dynamics and Earthquake Engineering 2007 (COMPDYN 2007)*, Rethymno, Crete, Greece, June 13-16.

Plevris, V., Lagaros, N. D., and Papadrakakis, M. (2008a). "Robust Seismic Optimum Design of Steel Structures." *3rd Panhellenic Conference on Earthquake Engineering and Engineering Seismology*, Athens, Greece, November 5-7 (in greek).

Plevris, V., Lagaros, N. D., and Papadrakakis, M. (2008b). "Structural Optimization based on the Particle Swarm Global Optimization Method." *8th World Congress on Computational Mechanics (WCCM8) - 5th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008)*, Venice, Italy, June 30 - July 5.

Plevris, V., and Papadrakakis, M. (2009a). "A Hybrid Particle Swarm - Gradient Algorithm for Global Structural Optimization." *Comput.-Aided Civ. Infrastruct. Eng.*, Submitted for publication.

Plevris, V., and Papadrakakis, M. (2009b). "A Hybrid Particle Swarm - SQP Scheme for the Optimum Design of Truss Structures." *8th World Conference on Structural and Multidisciplinary Optimization (WCSMO-8)*, Lisbon, Portugal, June 1-5.

Plevris, V., and Papadrakakis, M. (2009c). "Multi-objective Structural Optimization based on the Particle Swarm Optimization Method." *Computational Methods in*

Structural Dynamics and Earthquake Engineering 2009 (COMPDYN 2009), Rhodes, Greece, June 22-24.

Provatidis, C. G., and Venetsanos, D. T. (2006). "Cost minimization of 2D continuum structures under stress constraints by increasing commonality in their skeletal equivalents." *Forschung im Ingenieurwesen/Engineering Research*, 70(3), 159-169.

Pu, Y., Das, P. K., and Faulkner, D. (1997). "A strategy for reliability-based optimization." *Eng. Struct.*, 19 (3), 276-282.

Rackwitz, R. (2004). "Optimal and acceptable technical facilities involving risks." *Risk Analysis*, 24(3), 675-695.

Rackwitz, R., and Fiessler, B. (1978). "Structural Reliability under Combined Random Load Sequences." *Comput. Struct.*, 9, 484-494.

Rajashekhar, M. R., and Ellingwood, B. R. (1993). "A new look at the response surface approach for reliability analysis." *Struct. Safety*, 12(3), 205-220.

Ramm, E., Bletzinger, K.-U., Reitinger, R., and Maute, K. (1994). "The challenge of structural optimization." in *Advances in Structural Optimization*, B. H. V. Topping and M. Papadrakakis, eds., Civil-Comp Press, Edinburgh, 27-52.

Rangavajhala, S., Mullur, A., and Messac, A. (2007). "The challenge of equality constraints in robust design optimization: Examination and new approach." *Struct. Multidisc. Optim.*, 34(5), 381-401.

Ray, T., and Smith, W. (2006). "A surrogate assisted parallel multiobjective evolutionary algorithm for robust engineering design." *Engrg. Optim.*, 38(8), 997-1011.

Rechenberg, I. (1973). *Evolution strategy: optimization of technical systems according to the principles of biological evolution*, Frommann-Holzboog, Stuttgart.

Repalle, J., and Grandhi, R. V. (2005). "Reliability-based perform shape design in forging." *Communications in Numerical Methods in Engineering*, 21 (11), 607-617.

Reyes-Sierra, M., and Coello Coello, C. A. (2006). "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art." *International Journal of Computational Intelligence Research*, 2(3), 287-308.

Rizzi, P. (1976). "Optimization of multiconstrained structures based on optimality criteria." *17th Structures, structural dynamics and materials conference*, King of Prussia, PA, USA.

Rodriguez, R. J., and Montero, J. C. (2003). "Uncertainty characterisation reliability methods for slope stability." in *New Paradigms in Subsurface Prediction* M. S. Rosenbaum and A. K. Turner, eds., Springer Berlin / Heidelberg.

Rosenblueth, E., and Mendoza, E. (1971). "Reliability Optimization in Isostatic Structures." *Journal of the Engineering Mechanics Division ASCE*, 97(6), 1625-1642.

Rosenthal, R. E. (1985). "Principles of Multiobjective Optimization." *Decision Sciences*, 16, 133-152.

Ross, S. (2006). *Simulation*, 4th edn., Academic Press.

Royset, J. O., Der Kiureghian, A., and Polak, E. (2001). "Reliability-based optimal design of series structural systems." *J. Engrg. Mech.*, 127(6), 607-614.

Rubinstein, R. Y., and Kroese, D. P. (2008). *Simulation and the Monte Carlo Method*, 2nd edn., John Wiley & Sons, Inc.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). "Learning internal representations by error propagation." in *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, D. E. Rumelhart and J. L. McClelland, eds., MIT Press, Cambridge, 318-362.

Saarinen, S., Bramley, R., and Cybenko, G. (1993). "Ill-conditioning in neural network training problems." *SIAM Journal on Scientific Computing*, 14(3), 693-714.

Saliby, E. (1990). "Descriptive Sampling: A Better Approach to Monte Carlo Simulation." *Journal of the Operational Research Society*, 41(12), 1133-1142.

Saliby, E. (1997). "Descriptive Sampling: An improvement over Latin Hypercube Sampling." *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GE, USA, December 7-10.

Sandgren, E., and Cameron, T. M. (2002). "Robust design optimization of structures through consideration of variation." *Comput. Struct.*, 80, 1605-1613.

Sari, B., Amaitik, S., and Kilic, S. E. (2007). "A neural network model for the assessment of partners' performance in virtual enterprises." *Int. J. Adv. Manuf. Tech.*, 34(7-8), 816-825.

Sarma, K. C., and Adeli, H. (2000). "Fuzzy Genetic Algorithm for Optimization of Steel Structures." *J. Struct. Eng. (ASCE)*, 126(5), 596-604.

Sarma, K. C., and Adeli, H. (2002). "Life-cycle cost optimization of steel structures." *Int. J. Numer. Meth. Engng.*, 55(12), 1451-1462.

Schaffer, J. D. (1984). "Multiple objective optimization with vector evaluated genetic algorithms." PhD thesis, Vanderbilt University.

Schittkowski, K., Zillober, C., and Zotemantel, R. (1994). "Numerical comparison of nonlinear programming algorithms for structural optimization." *Struct. Optimization*, 7(1-2), 1-19.

Schmit, L. A., and Miura, H. (1976). "A New Structural Analysis/Synthesis Capability-ACCESS 1." *AIAA Journal*, 14(5), 661-671.

Schmit, L. A. J., and Farshi, B. (1974). "Some Approximation Concepts for Structural Synthesis." *AIAA Journal*, 12(5), 692-699.

Schoenauer, M. (1995). "Shape representation for evolutionary optimization and identification in structural mechanics." in *Genetic Algorithms in engineering and computer science*, G. Winter, J. Periaux, M. Galan, and P. Cuesta, eds., John Wiley & Sons, Chichester, UK, 443-464.

Schuëller, G. I. (2005). "Special Issue on Computational Methods in Stochastic Mechanics and Reliability Analysis." *Comput. Methods Appl. Mech. Engrg.*, 194(12-16), 1251-1795.

Schumacher, A., and Olschinka, C. (2008). "Robust design considering highly nonlinear structural behavior." *Struct. Multidisc. Optim.*, 35(3), 263-272.

Schwefel, H.-P. (1981). *Numerical optimization for computer models*, John Wiley & Sons, Chichester, UK.

Sexsmith, R. G. (1999). "Probability-based safety analysis - value and drawbacks." *Struct. Safety*, 21(4), 303-310.

Shao, R., Martin, E. B., Zhang, J., and Morris, A. J. (1997). "Confidence bounds for neural network representations." *Comput. Chem. Eng.*, 21(Supplement 1), S1173-S1178.

Shi, Y., and Eberhart, R. (1998). "A modified particle swarm optimizer." *IEEE World Congress on Computational Intelligence*, Anchorage, AK, USA, 69-73.

Shieh, R. C. (1994). "Massively parallel structural design using stochastic optimization and mixed neuralnet/finite element analysis methods." *Comput. Syst. Eng.*, 5(4-6), 455-467.

Shilov, G. E., and Gurevich, B. L. (1978). *Integral, Measure, and Derivative: A Unified Approach*, Dover Publications.

Song, B. F. (1997). "A technique for computing failure probability of a structure using importance sampling " *Comput. Struct.*, 62(4), 659-665.

Soong, T. T., and Grigoriou, M. (1993). *Random vibration of mechanical and structural systems*, Prentice Hall, Upper Saddle River, NJ.

Sorensen, J. D., and Tarp-Johansen, N. J. (2005). "Reliability-based optimization and optimal reliability level of offshore wind turbines." *International Journal of Offshore and Polar Engineering*, 15(2), 141-146.

Spears, W. M., and De Jong, K. A. (1990). "An Analysis of Multi-Point Crossover." *Proceedings of the Foundations of Genetic Algorithms Workshop*, 301-315.

Srinivasan, R. (2002). *Importance Sampling: Applications in Communications and Detection*, Springer-Verlag, Berlin.

Stadler, W. (1988). "Multicriteria Optimization in Engineering and in the Sciences." Plenum Press, New York.

Stephens, J. E., and VanLuchene, D. (1994). "Integrated assessment of seismic damage in structures." *Microcomputers in Civil Engineering*, 9(2), 119-128.

Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation, and Applications*, John Wiley & Sons.

Streicher, H., and Rackwitz, R. (2004). "Time-variant reliability-oriented structural optimization and a renewal model for life-cycle costing." *Probabilist. Eng. Mech.*, 19(1-2), 171-183.

Svanberg, K. (1987). "The method of moving asymptotes - a new method for structural optimization." *Int. J. Numer. Meth. Engng.*, 24(2), 359-373.

Szechtman, R. (2003). "Control variates techniques for Monte Carlo simulation." *Winter Simulation Conference Proceedings*, 144-149.

Taylor, C. A. (1989). *EQSIM, A program for generating spectrum compatible earthquake ground acceleration time histories, Reference Manual*, Bristol Earthquake Engineering Data Acquisition and Processing System.

Thanedar, P. B., Arora, J. S., Tseng, C. H., Lim, O. K., and Park, G. J. (1986). "Performance of some SQP algorithms on structural design problems." *Int. J. Numer. Meth. Engng.*, 23(12), 2187-2203.

Theocaris, P. S., and Panagiotopoulos, P. D. (1993). "Neural networks for computing in fracture mechanics. Methods and prospects of applications." *Comput. Methods Appl. Mech. Engrg.*, 106(1-2), 213-228.

Thierauf, G., and Cai, J. (1995). "A two level parallel evolution strategy for solving mixed-discrete structural optimization problems." *Proceedings of the 21th ASME Design Automation Conference*, Boston, MA, USA, September 17-22.

Thierauf, G., and Cai, J. (1996). "Structural optimization based on Evolution Strategy." in *Advanced computational methods in structural mechanics*, M. Papadrakakis and G. Bugeda, eds., CIMNE, Barcelona, 266-280.

Thompson, S. K. (2002). *Sampling*, 2nd edn., Wiley-Interscience.

Thompson, S. K., and Seber, G. A. F. (1996). *Adaptive Sampling*, John Wiley & Sons, New York.

Topping, B. H. V., and Bahreininejad, A. (1997). *Neural computing for structural mechanics*, Saxe-Coburg Publications, Edinburgh, UK.

Tsirigas, P., Gisakis, A., Plevris, V., and Papadrakakis, M. (2008). "Evaluation and Verification of the TRIC Shell Element." *6th GRACM International Congress on Computational Mechanics*, Thessaloniki, Greece, June 19-21.

Tsompanakis, Y., Lagaros, N. D., and Papadrakakis, M. (2008). "Structural optimization considering uncertainties." Structures and Infrastructures, D. M. Frangopol, ed., Taylor & Francis.

Van Keulen, F., and Hinton, E. (1996). "Topology design of plate and shell structures using the hard kill method." in *Advances in optimization for Structural Engineering*, B. H. V. Topping, ed., Civil-Comp Press, Edinburgh, 177-188.

Vanderplaats, G. N. (1984). *Numerical optimization techniques for engineering design: With applications*, McGraw-Hill, New York.

Venter, G., and Sobieszczanski-Sobieski, J. (2004). "Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization." *Struct. Multidisc. Optim.*, 26, 121-131.

Victoire, T. A. A., and Jeyakumar, A. E. (2004). "Hybrid PSO-SQP for economic dispatch with valve-point effect." *Electric Power Systems Research*, 71(1), 51-59.

Vlachogiannis, J. G., and Lee, K. Y. (2009). "Multi-objective based on parallel vector evaluated particle swarm optimization for optimal steady-state performance of power systems." *Expert. Syst. Appl.*(Article in Press, doi:10.1016/j.eswa.2009.02.079).

Waagen, D., Diercks, P., and McDonnell, J. (1992). "The stochastic direction set algorithm: A hybrid technique for finding function extrema." *Proceedings of the 1st Annual Conference on Evolutionary Programming*.

Wan, Y.-H. (1975). "On local Pareto optima." *Journal of Mathematical Economics*, 2(1), 35-42.

Waszczyszyn, Z., and Bartczak, M. (2002). "Neural prediction of buckling loads of cylindrical shells with geometrical imperfections." *Int. J. Non-linear Mech.*, 37(4-5), 763-775.

Watts, D. J. (1999). *Small Worlds: The Dynamics of Networks between Order and Randomness*, Princeton University Press.

Watts, D. J., and Strogatz, S. H. (1998). "Collective dynamics of 'small-world' networks." *Nature*, 393(6684), 440-442.

Weickum, G., Allen, M., and Maute, K. (2008). "Design optimization of stochastic dynamic systems by algebraic reduced order models." in *Structural design optimization considering uncertainties*, Y. Tsompanakis, N. D. Lagaros, and M. Papadrakakis, eds., Taylor & Francis, 135-154.

Werbos, P. J. (1974). "Beyond regression: New tools for prediction and analysis in the behavioral sciences." Harvard University, Cambridge, MA, USA.

Wolpert, D. H., and Macready, W. G. (1997). "No Free Lunch Theorems for Optimization." *Evol. Comput.*, 1(1).

Xie, Y. M., and Steven, G. P. (1993). "A simple evolutionary procedure for structural optimization." *Comput. Struct.*, 49(5), 885-896.

Ye, D., Chen, Z., and Liao, J. (2007). "A New Algorithm for Minimum Attribute Reduction Based on Binary Particle Swarm Optimization with Vaccination." in *Advances in Knowledge Discovery and Data Mining*, 1029-1036.

Yeniay, Ö. (2005). "Penalty Function Methods for Constraint Optimization with Genetic Algorithms." *Mathematical and Computational Applications*, 10(1), 45-56.

Youn, B. D., and Choi, K. K. (2004). "The performance moment integration method for reliability-based robust design optimization." *Proceedings of the ASME Design Engineering Technical Conference 1*, 915-925.

Youn, B. D., Choi, K. K., Du, L., and Gorsich, D. (2007). "Integration of possibility-based optimization and robust design for epistemic uncertainty." *J. Mech. Des. - T. ASME*, 129(8), 876-882.

Zacharias, J., Hartmann, C., and Delgado, A. (2004). "Damage detection on crates of beverages by artificial neural networks trained with finite-element data." *Comput. Methods Appl. Mech. Engrg.*, 193(6-8), 561-574.

Zang, C., Friswell, M. I., and Mottershead, J. E. (2005). "A review of robust optimal design and its application in dynamics." *Comput. Struct.*, 83(4-5), 315-326.

Zeleny, M. (1982). *Multiple Criteria Decision Making*, McGraw-Hill, New York.

Zhang, J.-R., Zhang, J., Lok, T.-M., and Lyu, M. R. (2007). "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training." *Appl. Math. Comput.*, 185, 1026-1037.

Zhou, M., and Rozvany, G. I. N. (1993). "DCOC: An optimality criteria method for large systems. Part II: Algorithm." *Struct. Optimization*, 6, 250-262.

Zitzler, E. (1999). "Evolutionary algorithms for multiobjective optimization: methods and applications." PhD thesis, Swiss Federal Institute of Technology Zurich.

Zitzler, E., Deb, K., and Thiele, L. (2000). "Comparison of multiobjective evolutionary algorithms: empirical results." *Evol. Comput.*, 8(2), 173-195.

Appendix A.
Notation and Symbols

This appendix contains the details of the mathematical notation used in the thesis. A notation based on ISO 31-11 (ISO 1993) is mainly followed, with some exceptions and additions. The following general rules are applied:

- The *International System of Units* (SI) base units as well as the SI derived units are basically used, unless otherwise stated.
- Italic letters are used for variables (a, x), parameters or functions (f, g), while vectors and matrices are denoted by boldface letters, lowercase ($\boldsymbol{x}$, $\boldsymbol{r}$) and uppercase ($\boldsymbol{A}$, $\boldsymbol{M}$), respectively.
- Physical quantities consist of a numeral times a unit. These are typeset following the recommendations of ISO 1000 (ISO 1993). The units (and their prefixes) as well as the numbers are printed in upright Roman type, so as to differentiate from the italic type used for variables (m for mass, l for length), while a thin space (narrower than that between words) separates the number and the symbol (2.2 kg, 5 kN). This rule explicitly includes the percent sign.
- Multiplication between numbers in the numerical value of a physical quantity is indicated by a "×" while symbols for derived units formed from multiple units by multiplication are joined with a thin space or center dot (·), for example "5×10^3 kN m" or "5×10^3 kN·m.
- A thin space is used as a thousand separator in contrast to commas or periods in order to reduce confusion (10 000 kPa).
- Vector or matrix components are denoted with the index notation. In this notation, u_i is the i-th component of vector $\boldsymbol{u}$, while $M_{i,j}$ is the i,j-th component of matrix $\boldsymbol{M}$.
- Individual vectors that belong to a set of vectors are denoted using a superscript, for example vector $\boldsymbol{v}^j$, $j=1, \ldots, n$, in order to avoid confusion with vector elements that are denoted with subscripts, as described above.
- The index notation is also used for shorthand notation of partial derivatives, for example $\partial_t f(\boldsymbol{u},t) := \partial f(\boldsymbol{u},t) / \partial t$ is the partial derivative of the function $f(\mathbf{u},t)$ with respect to t.

The lists below present analytically the mathematical notation used in the thesis.

i. Mathematical Logic

Symbol	Name	Meaning / Example
$\wedge$	Conjunction sign	$p \wedge q$ means "p and q"
$\vee$	Disjunction sign	$p \vee q$ means "p or q (or both)"
$\neg$	Negation sign	The statement $\neg\, p$ means "negation of p"; "not p"; "non p". It is true if and only if p is false
$\Rightarrow$	Implication sign	$p \Rightarrow q$ means "if p then q"; "p implies q"
$\forall$	Universal quantifier	For all, for every
$\exists$	Existential quantifier	There is, there exists

ii. Sets

Symbol	Meaning	Example and verbal equivalent
$\in$	Belongs to; is an element of	$x \in A$ means "x belongs to A"
$\notin$	Does not belong to; is not an element of	$x \notin A$ means "x does not belong to A"
$\ni$	Contains	$A \ni x$ means "set A contains x" (as an element)
$\not\ni$	Does not contain	$A \not\ni x$ means "set A does not contain x" (as an element) $A \not\ni x \Leftrightarrow x \notin A$
{ ... }	Set with elements	$\{x_1, x_2, x_3\}$ means "a set with elements x_1, x_2, x_3"
{ ... | ... }	Set with elements for which a proposition is true	$\{x \in A \mid p(x)\}$ means a set of those elements of A for which the proposition $p(x)$ is true
card	Number of elements in a set; cardinal of a set	card(A) means the number of elements in A
[... , ...]	Closed interval in $\mathbb{R}$	$[a,b] = \{x \in \mathbb{R} \mid a \le x \le b\}$
(... , ...)	Open interval in $\mathbb{R}$	$(a,b) = \{x \in \mathbb{R} \mid a < x < b\}$
(... , ...]	Left half-open interval in $\mathbb{R}$	$(a,b] = \{x \in \mathbb{R} \mid a < x \le b\}$
[... , ...)	Right half-open interval in $\mathbb{R}$	$[a,b) = \{x \in \mathbb{R} \mid a \le x < b\}$
$\subseteq$	Is included in	$B \subseteq A$ means "B is included in A"; "B is a subset of A"; "every element of B belongs to A"
$\subset$	Is properly included in	$B \subset A$ means "B is properly included in A"; "B is a proper subset of A"; "every element of B belongs to A, but B is not equal to A"

$\nsubseteq$	Is not included in	$C \nsubseteq A$ means "C is not included in A"; "C is not a subset of A"
$\supseteq$	Includes	$A \supseteq B$ means "A includes B (as subset)"; "A contains every element of B"
$\supset$	Includes properly	$A \supset B$ means "A includes B properly"; "A contains every element of B, but A is not equal to B"
$\not\supseteq$	Does not include	$A \not\supseteq C$ means "A does not include C (as subset)" $A \not\supseteq C \Leftrightarrow C \not\subseteq A$
$\cup$	Union of two sets	$A \cup B$ means "Union of A and B", i.e. "the set of elements which belong to A or to B or to both A and B" $A \cup B = \{ x \mid x \in A \vee x \in B \}$
$\cap$	Intersection of two sets	$A \cap B$ means "Intersection of A and B", i.e. "the set of elements which belong to both A and B" $A \cap B = \{ x \mid x \in A \wedge x \in B \}$
$\setminus$	Difference of two sets; one set minus another	$A \setminus B$ means "the set of elements which belong to A but not to B" $A \setminus B = \{ x \mid x \in A \wedge x \notin B \}$
$\varnothing$	Empty set	The empty set is an implicit member of every set
$\mathbb{N}$	The set of natural numbers; the set of positive integers and zero	$\mathbb{N} = \{0,1,2,3,\ldots\}$
$\mathbb{Z}$	The set of integer numbers	$\mathbb{Z} = \{\ldots,-3,-2,-1,0,1,2,3,\ldots\}$
$\mathbb{Q}$	The set of rational numbers	
$\mathbb{R}$	The set of real numbers	
$\mathbb{R}^n$	Euclidean n-dimensional real vector space	Set of all n-dimensioned vectors with real entries
$\mathbb{R}^{m\times n}$	Euclidean $m\times n$-dimensional real matrix space	Set of all $m\times n$ matrices with real entries
$\mathbb{C}$	The set of complex numbers	Set of numbers consisting of a real and an imaginary part
$\mathbb{C}^n$	Euclidean n-dimensional complex vector space	Set of all n-dimensioned vectors with complex entries
$\mathbb{C}^{m\times n}$	Euclidean $m\times n$-dimensional complex matrix space	Set of all $m\times n$ matrices with complex entries

iii. Miscellaneous signs and symbols

Symbol	Meaning	Example / verbal equivalent
:=	Is by definition equal to	$a := b$ means "a is by definition equal to b"
=	Is equal to	$a = b$ means "a is equal to b"
≠	Is not equal to	$a \neq b$ means "a is not equal to b"
≈	Is approximately equal to	$a \approx b$ means "a is approximately equal to b"
≃	Is asymptotically equal to	$a \simeq b$ means "a is asymptotically equal to b"
$\triangleq$	Corresponds to	On a map: 1cm $\triangleq$ 1km
~	Is proportional to	$a \sim b$ means "a is proportional to b"
<	Is less than *	$a < b$ means "a is less than b"
>	Is greater than *	$a > b$ means "a is greater than b"
≤	Is less than or equal to *	$a \leq b$ means "a is less than or equal to b"
≥	Is greater than or equal to *	$a \geq b$ means "a is greater than or equal to b"
≪	Is much less than *	$a \ll b$ means "a is much less than b"
≫	Is much greater than *	$a \gg b$ means "a is much greater than b"
∞	Infinity	$\lim_{x \to \infty} f(x) = \infty$ means "the limit of function $f(x)$ as x approaches infinity is infinity"
||	Is parallel to	AB || CD means "the line AB is parallel to the line CD"
⊥	Is perpendicular to	AB ⊥ CD means "the line AB is perpendicular to the line CD"
$\boldsymbol{I}$	Identity matrix	Matrix containing ones at its diagonal and zeros everywhere else
$\boldsymbol{X}^{\mathrm{T}}$	Transpose of matrix $\boldsymbol{X}$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
$\boldsymbol{X}^{-1}$	Inverse of matrix $\boldsymbol{X}$	$\begin{bmatrix} 2 & 4 \\ 3 & 7 \end{bmatrix}^{-1} = \begin{bmatrix} 3.5 & -2 \\ -1.5 & 1 \end{bmatrix}$
$\sqrt{x}$	Positive square root of scalar x	$\sqrt{4} = 2$
$\lvert x \rvert$	Absolute value of scalar x **	$\lvert -5 \rvert = 5$
$\operatorname{sgn}(x)$	Signum of scalar x **	$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$
$\lVert \boldsymbol{x} \rVert$	Euclidean norm of vector $\boldsymbol{x}$	$\lVert \boldsymbol{x} \rVert := \sqrt{\boldsymbol{x}^{\mathrm{T}} \cdot \boldsymbol{x}} = \sqrt{x_1^2 + \ldots + x_n^2}$

∂	Partial derivative	$\partial_t f(\boldsymbol{u},t) := \partial f(\boldsymbol{u},t) / \partial t$
∇	Gradient of	∇f is the gradient vector of $f(\boldsymbol{x})$
∇^2	Hessian of	$\nabla^2 f$ is the hessian matrix of $f(\boldsymbol{x})$

* When used for comparison of vectors or matrices instead of scalars, it means entrywise comparison of vectors or matrices, i.e. for vectors $\boldsymbol{a}$, $\boldsymbol{b}$, the expression $\boldsymbol{a} < \boldsymbol{b}$ means $\boldsymbol{a}_i < \boldsymbol{b}_i$ for every i.

** See paragraph vii (Complex numbers) for corresponding definition for complex numbers.

iv. Operations

Symbol	Meaning	Example / verbal equivalent
+	Addition ("plus")	2+5=7
-	Subtraction ("minus")	8-2=6
· or ×	Multiplication ("times") for scalars, vectors or matrices	$5 \cdot 3 = 15$ $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 11 & 4 \end{bmatrix}$
/	Division ("divided by")	
$\pm$	Plus *or* minus	$a \pm b$ means "a plus or minus b"
$\mp$	Minus *or* Plus	$a \mp b$ means "a minus or plus b" $-(a \pm b) = a \mp b$
$\circ$	Hadamard product (entry-wise product or Schur product) for vector or matrix multiplication	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 6 & 4 \end{bmatrix}$

v. Variables

Style	Example	Meaning
italics	x, a, n, i, t	Scalar variable or
	f, g, h	Scalar function
Boldface italics	$\boldsymbol{x}$, $\boldsymbol{a}$, $\boldsymbol{r}$, $\boldsymbol{s}$, $\boldsymbol{t}$	Vector variable or
	$\boldsymbol{f}$, $\boldsymbol{g}$, $\boldsymbol{h}$	Vector function
Upright	cos, sin, max, min d (differential operator)	Explicitly defined functions and operators
Boldface uppercase italics	$\boldsymbol{X}$, $\boldsymbol{A}$, $\boldsymbol{R}$, $\boldsymbol{S}$, $\boldsymbol{T}$	Matrix variable

Boldface number	$\mathbf{0}$, $\mathbf{1}$	Vector or matrix containing the specific number for every element
Underlined italics	$\underline{x}$, $\underline{a}$, $\underline{r}$, $\underline{s}$, $\underline{t}$	Scalar random variable (Hemelrijk 1966)
Boldface underlined italics	$\underline{\boldsymbol{x}}$, $\underline{\boldsymbol{a}}$, $\underline{\boldsymbol{r}}$, $\underline{\boldsymbol{s}}$, $\underline{\boldsymbol{t}}$	Vector random variable
Boldface uppercase underlined italics	$\underline{\boldsymbol{X}}$, $\underline{\boldsymbol{A}}$, $\underline{\boldsymbol{R}}$, $\underline{\boldsymbol{S}}$, $\underline{\boldsymbol{T}}$	Matrix random variable
Capital Calligraphic	$\mathcal{F}$, $\mathcal{A}$, $\mathcal{C}$	Set, Space
Subscript	$\boldsymbol{x}=[x_1, \ldots, x_n]$	Vector components or
	$\boldsymbol{M}=\begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}$	Matrix components or
	$\vartheta_t f(\boldsymbol{u},t) := \vartheta f(\boldsymbol{u},t) / \vartheta t$	Partial derivatives

vi. Constants

Symbol	Meaning	Approximate value
π	Ratio of the circumference of a circle to its diameter	π ≈ 3.1415926
e	Euler's number, base of natural logarithms	e ≈ 2.71828 18

vii. Complex numbers

Symbol	Meaning	Example and verbal equivalent
$\imath$	Imaginary unit	$\imath^2 = -1$
Re(z)	Real part of a complex number z	Re(x+$y\imath$)=x
Im(z)	Imaginary part of z	Im(x+$y\imath$)=y
$\lvert z \rvert$	Absolute value of z	$\lvert x + y\imath \rvert = x^2 + y^2$
arg(z)	Argument of z; the angle between the positive real axis and the vector representing z on an Argand diagram.	$z = x + y\imath = r\cos(\varphi) + \imath r\sin(\varphi)$ where: $r=\lvert z \rvert$ and φ=arg(z)
$\overline{z}$	Conjugate of z	$\overline{x + y\imath} = \overline{x - y\imath}$
sgn(z)	Signum of z	sgn(z)=z/|z|, for z≠0 sgn(z)=0, for z=0

viii. Special notation for multi-objective optimization

Symbol	Meaning	Example and verbal equivalent
$\succeq$	Weakly dominates	$\boldsymbol{u} \succeq \boldsymbol{v}$ means "objective vector *u* weakly dominates objective vector *v*"
$\succ$	Dominates	$\boldsymbol{u} \succ \boldsymbol{v}$ means "objective vector *u* dominates objective vector *v*"
$\succ\!\!\succ$	Strictly dominates	$\boldsymbol{u} \succ\!\!\succ \boldsymbol{v}$ means "objective vector *u* strictly dominates objectivess vector *v*"
$\preceq$	Is weakly dominated by	$\boldsymbol{u} \preceq \boldsymbol{v}$ means "objective vector *u* is weakly dominated by objective vector *v*"
$\prec$	Is dominated by	$\boldsymbol{u} \prec \boldsymbol{v}$ means "objective vector *u* is dominated by objective vector *v*"
$\prec\!\!\prec$	Is strictly dominated by	$\boldsymbol{u} \prec\!\!\prec \boldsymbol{v}$ means "objective vector *u* is strictly dominated by objective vector *v*"
$\prec\!\succ$	Is incomparable to (or indifferent to)	$\boldsymbol{u} \prec\!\succ \boldsymbol{v}$ means "objective vector *u* is incomparable to objective vector *v*"

ix. Statistical operators

Symbol	Meaning	Example and verbal equivalent
$\mathbb{E}(\underline{x})$	Expected value operator	$\mathbb{E}(\underline{x}) = \int_{\mathbb{R}} x \cdot f(x)\mathrm{d}x$ is the mean value, or expected value of random variable $\underline{x}$ with Probability Density Function $f(x)$
$\mathrm{var}(\underline{x})$	Variance operator	$\mathrm{var}(\underline{x}) = \mathbb{E}\left((\underline{x} - \mu)^2 \right)$ is the variance of random variable $\underline{x}$ with $\mu = \mathbb{E}(\underline{x})$
$\mathrm{cov}(\underline{x}, \underline{y})$	Covariance operator	$\mathrm{cov}(\underline{x}, \underline{y}) = \mathbb{E}\left((\underline{x} - \mu) \cdot (\underline{x} - \nu) \right)$ is the covariance of random variables $\underline{x}$ and $\underline{y}$ with $\mu = \mathbb{E}(\underline{x})$ and $\nu = \mathbb{E}(\underline{y})$

Appendix B.
Acronyms and Abbreviations

This appendix contains an alphabetized listing of the acronyms and abbreviations used in the text, figures and tables of the dissertation.

i. Acronyms

Acronyms in text are spelled out the first time that they appear in each chapter, with the shortened form appearing immediately in parentheses. Thereafter, the shortened form is used throughout the chapter.

Acronym	Description
AI	**A**rtificial **I**ntelligence
BFGS	**B**royden-**F**letcher-**G**oldfarb-**S**hanno
BPNN	**B**ack-**P**ropagation **N**eural **N**etwork
CDF	**C**umulative **D**istribution **F**unction
CEA	**C**ascade **E**volutionary **A**lgorithm
CHS	**C**ircular **H**ollow **S**ection
CM	**C**onstraint **M**ethod
CPU	**C**entral **P**rocessing **U**nit
CSA	**C**onventional **S**emi-**A**nalytical
DDO	**D**eterministic **D**esign **O**ptimization
DFM	**D**istance **F**unction **M**ethod
DM	**D**ecision **M**aker
DOF	**D**egree **O**f **F**reedom
DS	**D**escriptive **S**ampling
DTI	**D**irect **T**ime **I**ntegration
EA	**E**volutionary **A**lgorithm
EAS	**E**qual **A**ngle **S**ection
EP	**E**volutionary **P**rogramming
ES	**E**volution **S**trategy or **E**volution **S**trategies
ESA	**E**xact **S**emi-**A**nalytical
ESMO	**E**volution **S**trategy for **M**ulti-objective **O**ptimization
FEA	**F**inite **E**lement **A**nalysis
FEM	**F**inite **E**lement **M**ethod
FLS	**F**atigue **L**imit **S**tate
FORM	**F**irst-**O**rder **R**eliability **M**ethod

GA	**G**enetic **A**lgorithm
GFD	**G**lobal **F**inite **D**ifference
GP	**G**oal **P**rogramming
GRG	**G**eneralized **R**educed **G**radient
HSS	**H**ammersley **S**equence **S**ampling
IS	**I**mportance **S**ampling
KKT	**K**arush-**K**uhn-**T**ucker
LHS	**L**atin **H**ypercube **S**ampling
LMS	**L**east **M**ean **S**quares
LWM	**L**inear **W**eighting **M**ethod
MCS	**M**onte **C**arlo **S**imulation
MLHS	**M**edian **L**atin **H**ypercube **S**ampling
MMA	**M**ethod of **M**oving **A**symptotes
MOP	**M**ulti-objective **O**ptimization **P**roblem
MP	**M**athematical **P**rogramming
MPP	**M**ost **P**robable **P**oint
MSE	**M**ean **S**quare **E**rror
NFL	**N**o **F**ree **L**unch
NLP	**N**on-**L**inear **P**rogramming
NN	**N**eural **N**etwork
PDF	**P**robability **D**ensity **F**unction
PF	**P**areto **F**ront
PSO	**P**article **S**warm **O**ptimization
PVM	**P**arallel **V**irtual **M**achine
QP	**Q**uadratic **P**rogramming
RBDO	**R**eliability-**B**ased **D**esign **O**ptimization
RDO	**R**obust **D**esign **O**ptimization
RNN	**R**ecurrent **N**eural **N**etwork
ROM	**R**educed **O**rder **M**odel
Rprop	**R**esilient **prop**agation
RRDO	**R**eliability-based **R**obust **D**esign **O**ptimization
RSM	**R**esponse **S**urface **M**ethod
RSMA	**R**esponse **S**pectrum **M**odal **A**nalysis
SA	**S**emi-**A**nalytical
SI	Le **S**ystème **I**nternational d'Unités (French) International System of Units (English)
SLP	**S**equential **L**inear **P**rogramming
SLS	**S**erviceability **L**imit **S**tate
SOP	**S**ingle-objective **O**ptimization **P**roblem
SORM	**S**econd-**O**rder **R**eliability **M**ethod
SQP	**S**equential **Q**uadratic **P**rogramming
SRSS	**S**quare **R**oot of the **S**um of **S**quares
TRM	**T**rust-**R**egion **M**ethod

ULS	**U**ltimate **L**imit **S**tate
WQM	**W**eighted **Q**uadratic **M**ethod
XOR	E**x**clusive **OR**

ii. Abbreviations

Abbreviation	**Description**
a.k.a.	**a**lso **k**nown **a**s
e.g.	**e**xempli **g**ratia (Latin)
	for sake of example, for example (English)
ed.	**ed**itor
Eq.	**eq**uation
et al.	**et al**ii (Latin)
	and others (English)
etc.	**et c**etera (Latin)
	and so forth (English)
i.e.	**i**d **e**st (Latin)
	that is (English)
no.	**n**umer**o** (Latin)
	number (English)
p.	**p**age

Appendix C.
Listing of Publications

This appendix contains a listing of publications by the author, in chronological order, from the newest to the oldest, for every category. Twenty nine (29) publications have been made in total, divided in three categories:

i. Nine (9) publications in referred international scientific journals (2 under review);

ii. Two (2) scientific book chapters;

iii. Eighteen (18) publications in international scientific conference proceedings.

For every work published in international scientific journals, the total number of citations is reported. The number of citations was taken from two sources on June 5, 2009: (i) **Scopus**® (www.scopus.com); and (ii) **Google Scholar** (scholar.google.com). Scopus® is a registered trademark of Elsevier BV. Google Scholar is a service provided by Google.

i. Publications in referred international scientific journals

1. **V. Plevris** and M. Papadrakakis, "A Hybrid Particle Swarm – Gradient Algorithm for Global Structural Optimization", *Computer-Aided Civil and Infrastructure Engineering*, submitted for publication, 2009 (Plevris and Papadrakakis 2009a).
2. N. D. Lagaros, **V. Plevris** and M. Papadrakakis, "Neurocomputing Strategies for Solving Reliability-Robust Design Optimization Problems", *Engineering Computations*, submitted for publication, 2009 (Lagaros et al. 2009).
3. N. D. Lagaros, **V. Plevris** and M. Papadrakakis, "Reliability Based Robust Design Optimization of Steel Structures", *Int. J. Simul. Multidisci. Des. Optim.*, 1(1), 19–29, 2007 (Lagaros et al. 2007).
4. M. Papadrakakis, N.D. Lagaros and **V. Plevris**, "Design Optimization of Steel Structures Considering Uncertainties", *Eng. Struct.*, 27(9), 1408-1418, 2005 (Citations: Scopus 13, Google scholar 12) (Papadrakakis et al. 2005).
5. N. D. Lagaros, **V. Plevris** and M. Papadrakakis, "Multi-objective Design Optimization Using Cascade Evolutionary Computations", *Comput. Methods Appl. Mech. and Engrg.*, 194(30-33), 3496-3515, 2005 (Citations: Scopus 12, Google scholar 14) (Lagaros et al. 2005c).

6. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, "Structural Optimization Considering the Probabilistic System Response", *Theoret. Appl. Mech.*, 31(3-4), 361-394, 2004 (Citations: Google scholar 3) (Papadrakakis et al. 2004a).
7. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, "Multi-objective Optimization of Skeletal Structures under Static and Seismic Loading Conditions", *Engrg. Optim.*, 34(6), 645-669, 2002 (Citations: Scopus 7, Google scholar 13) (Papadrakakis et al. 2002a).
8. M. Papadrakakis, N. D. Lagaros, Y. Tsompanakis and **V. Plevris**, "Large Scale Structural Optimization: Computational Methods and Optimization Algorithms", *Archives of Computational Methods in Engineering*, 8(3), 239-301, 2001 (Citations: Scopus 12, Google scholar 15) (Papadrakakis et al. 2001b).
9. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, 'Optimum Design of Space Frames under Seismic Loading', *Int. J. Struct. Stabil. Dyn.*, 1(1), 105-123, 2001 (Citations: Scopus 2, Google scholar 3) (Papadrakakis et al. 2001a).

ii. Scientific book chapters

1. N. D. Lagaros, Y. Tsompanakis, M. Fragiadakis, **V. Plevris** and M. Papadrakakis, "Metamodel-based Computational Techniques for Solving Structural Optimization Problems Considering Uncertainties", in *Structural Design Optimization Considering Uncertainties*, Y. Tsompanakis, N.D. Lagaros and M. Papadrakakis (Eds.), Taylor and Francis (ISBN: 978-0-415-45260-1), 567-597, 2008 (Lagaros et al. 2008b).
2. N. D. Lagaros, M. Papadrakakis and **V. Plevris**, "Multi-objective Optimization of Space Structures under Static and Seismic Loading Conditions", in *Evolutionary Multiobjective Optimization*, A. Abraham, L. C. Jain and R. Goldberg (Eds.), Springer (ISBN: 1-85233-787-7), 273-300, 2005 (Lagaros et al. 2005b).

iii. Publications in international scientific conference proceedings

1. **V. Plevris** and M. Papadrakakis, "Multi-objective Structural Optimization based on the Particle Swarm Optimization Method", *Computational Methods in Structural Dynamics and Earthquake Engineering 2009 (COMPDYN 2009)*, Rhodes, Greece, June 22-24, 2009 (Plevris and Papadrakakis 2009c).
2. A. Gisakis, P. Tsirigas, **V. Plevris** and M. Papadrakakis, "Evaluation Study of the Improved TRIC Shell Element for Linear and Nonlinear Structural Analysis", *2nd South-East European Conference on Computational Mechanics (SEECCM 2009)*, Rhodes, Greece, June 22-24, 2009 (Gisakis et al. 2009).
3. **V. Plevris** and M. Papadrakakis, "A Hybrid Particle Swarm - SQP Scheme for the Optimum Design of Truss Structures", *8th World Conference on Structural and Multidisciplinary Optimization (WCSMO-8)*, Lisbon, Portugal, June 1-5, 2009 (Plevris and Papadrakakis 2009b).

4. **V. Plevris**, N. D. Lagaros and M. Papadrakakis, "Robust Seismic Optimum Design of Steel Structures", *3rd Panhellenic Conference on Earthquake Engineering and Engineering Seismology*, Athens, Greece, November 5-7, 2008 (in greek) (Plevris et al. 2008a).
5. **V. Plevris**, N. D. Lagaros and M. Papadrakakis, "Structural Optimization based on the Particle Swarm Global Optimization Method", *8th World Congress on Computational Mechanics (WCCM8) - 5th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008)*, Venice, Italy, June 30 – July 5, 2008 (Plevris et al. 2008b).
6. P. Tsirigas, A. Gisakis, **V. Plevris** and M. Papadrakakis, "Evaluation and Verification of the TRIC Shell Element", *6th GRACM International Congress on Computational Mechanics*, Thessaloniki, Greece, June 19-21, 2008 (Tsirigas et al. 2008).
7. **V. Plevris**, N. D. Lagaros and M. Papadrakakis, "Seismic Design Optimization of Steel Structures Considering Uncertainties", *Computational Methods in Structural Dynamics and Earthquake Engineering 2007 (COMPDYN 2007)*, Rethymno, Crete, Greece, June 13-16, 2007 (Plevris et al. 2007).
8. **V. Plevris**, N. D. Lagaros and M. Papadrakakis, "Advances in Evolutionary Structural Optimization Considering Uncertainties", *7th World Congress on Computational Mechanics (WCCM VI)*, Los Angeles, California, USA, July 16-22, 2006 (Plevris et al. 2006b).
9. M. Papadrakakis, N. D. Lagaros, **V. Plevris**, D.C. Charmpis and M. Fragiadakis, "Computational Challenges in Optimum Structural Design", *7th World Congress on Computational Mechanics (WCCM VII)*, Los Angeles, California, USA, July 16-22, 2006 (Papadrakakis et al. 2006).
10. **V. Plevris**, N. D. Lagaros, D.C. Charmpis and M. Papadrakakis, "Metamodel Assisted Techniques for Structural Optimization", *First South-East European Conference on Computational Mechanics (SEECCM 06)*, Kragujevac, Serbia, June 28-30, 2006 (Plevris et al. 2006a).
11. **V. Plevris**, N. D. Lagaros and M. Papadrakakis, "Robust Design Optimization of Steel Structures Using Cascade Evolutionary Computations", *Eighth U.S. National Congress on Computational Mechanics (USNCCM 8)*, Austin Texas, USA, July 24-28, 2005 (Plevris et al. 2005b).
12. **V. Plevris**, N. D. Lagaros and M. Papadrakakis, "Robust Design Optimization of Steel Structures", *5th GRACM International Congress on Computational Mechanics, Limassol*, Cyprus, June 29 – July 1, 2005 (Plevris et al. 2005a).
13. N. D. Lagaros, **V. Plevris**, Y. Tsompanakis and M. Papadrakakis, "Multi-performance Robust Optimum Design of Steel Structures", *6th World Congress on Structural and Multidisciplinary Optimization (WCSMO 6)*, Rio de Janeiro, Brazil, May 30 – June 3, 2005 (Lagaros et al. 2005d).

14. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, "Structural Optimization Considering the Probabilistic System Response", *The First International Conference on Computational Mechanics (CM'04)*, Belgrade, Serbia, November 15-17, 2004 (Papadrakakis et al. 2004b).

15. M. Papadrakakis, **V. Plevris**, N. D. Lagaros and V. Papadopoulos, "Robust Design Optimization of 3D Truss Structures Using Evolutionary Computation", *6th World Congress on Computational Mechanics (WCCM VI) in conjunction with APCOM'04*, Beijing, China, September 5-10, 2004 (Papadrakakis et al. 2004c).

16. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, "Multi-objective Optimum Design of 3D Structures under Static and Seismic Loading Conditions", *4th GRACM Congress on Computational Mechanics*, Patra, Greece, June 27-29, 2002 (Papadrakakis et al. 2002b).

17. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, "Optimum Design of Structures under Seismic Loading", *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2000)*, Barcelona, Spain, September 11-14, 2000 (Papadrakakis et al. 2000b).

18. M. Papadrakakis, N. D. Lagaros and **V. Plevris**, "Optimum Design of 3D Structures under Static and Dynamic Loading Conditions", *Fourth International Colloquium on Computation of Shell & Spatial Structures (IASS-IACM 2000)*, Chania, Greece, June 4-7, 2000 (Papadrakakis et al. 2000a).