A Design and Optimization Assistant for Induction Motors and Generators

by

Ujjwal Sinha
B. Tech., Indian Institute of Technology, Delhi, India (1991)
S.M., Massachusetts Institute of Technology (1993)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

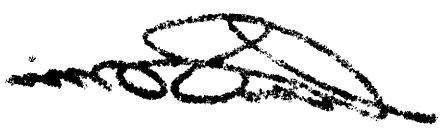
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author.....
Department of Mechanical Engineering
February 18, 1998

Certified by.....
James L. Kirtley Jr.
Professor, Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by.....
Ain A. Sonin
Chairman, Departmental Committee on Graduate Students



A Design and Optimization Assistant for Induction Motors and Generators

by
Ujjwal Sinha
Department of Mechanical Engineering

Submitted to the Department of Mechanical Engineering
on February 18, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis presents a design methodology and software design tool, which are useful for the design of induction motors and synchronous generators. A user or designer specifies performance requirements and the system synthesizes a set of design parameters which meet those specifications. Optimization may also be performed by the designer with respect to any performance parameter, while keeping other requirements within specified limits.

Electric machine design is in general a "hard" problem, and most designers rely on their knowledge, experience, and intuition to design new motors or modify existing ones. Most of the problems encountered can be traced to non-linearities, coupled equations, categorical variables, and presence of multiple objectives. Analysis of given design variables to compute performance parameters is comparatively easier using circuit equation analysis routines. The converse (synthesis process), where we need to generate a set of design variables matching certain performance criteria, is a much harder problem. This is also the more common problem in a design scenario. We propose a two-step methodology to generate designs matching user requirements, and perform optimizations.

In the first step of our methodology, a Monte-Carlo based statistical approach is proposed to circumvent the aforementioned problems. The n-dimensional design space is first reduced to a smaller sub-space which is more likely to contain the desired solutions. A multivariate normal distribution is used to characterize this sub-space. Several designs are generated within this sub-space which allow a user to evaluate multiple design possibilities. All of these designs meet user requirements.

These designs are then also used as starting points for further optimization, in the second step of our methodology. A statistical function approximation tool called MARS (MultiVariate Adaptive Regression Splines) is used to "map" the relations between inputs and every performance variable. This map is then used during the optimization process for obtaining function values and gradients at all locations. A non-linear programming algorithm is used to perform all optimizations. Ideas from multiple objective optimization literature are used to account for multiple performance variables.

The proposed methodology is implemented in an industrial strength software system which allows a firm to perform multiple scenario analyses, automate the design process, perform optimizations, shorten development lead times, and react fast to customer requests. Several examples using industrial strength circuit analysis routines are presented, and their

results analyzed.

Even though this approach is applied to the case of induction motors, and synchronous generators, it is believed that the methodology is sufficiently general, and would be applicable to many design situations.

Thesis Committee:

Woodie C. Flowers, Committee Chairman
Pappalardo Professor of Mechanical Engineering

Roy E. Welsch, Committee Co-Chair
Professor of Statistics and Management Sciences

James L. Kirtley Jr., Thesis Supervisor
Professor of Electrical Engineering

In the memory of my mother

Premi Sinha (1938–1986)

who would have been really happy to see this thesis.

Acknowledgments

First and foremost, I would like to thank my thesis supervisor Professor James L. Kirtley Jr. for all his support and guidance throughout the project. Without his encouragement and inspiration at every stage, this project could not have been completed.

I would also like to thank the chairman and co-chair of my thesis committee: Professor Woodie Flowers, and Professor Roy Welsch, for their insightful comments, and helpful suggestions.

This thesis was supported by MagneTek Inc. Sincere thanks are due to Paul Lindhorst, for his help and support. To Bob Oesterlei, for helping and guiding me at every stage of the project over the last 5 years, and for being my virtual "co-advisor" on the project. To Bob Lambrecht, Chris White, Gail Sahlin, and Ed Eden for providing the software support and putting up with all my questions and requests!

Sincere thanks are due to Atul Adya, a close friend, for his support and help with the programming aspects of this project. To Karen, Vivian, and the staff and students of the Laboratory for Electromagnetic and Electronic Systems (LEES), I say a collective thank you.

I wish to thank Leslie Regan from the Mechanical Engineering Graduate Office and Dean Danielle Guichard–Ashbrook (International Students Office) who went out of their way to help me out during my first few days at MIT.

Most of all, thanks to my wife Shefali for all the love, care, and affection, and for being really understanding when the going was tough. And to my father and brother, whose constant support and encouragement have made a dream come true. To my entire family, I owe a debt of love and gratitude that I will never be able to repay.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation for the Project

Industry today is faced with a variety of challenges in a fast changing world. Competitive pressures have forced companies to reduce the time to market their products, react fast to customer requirements, innovate continuously, and tailor their products to add value for customers. Firms today are expected to deliver high quality, low cost, innovative products with extremely short development lead times. Rapidly changing technology has also significantly changed the way businesses are managed. Careful attention has to be paid to managing information and enhancing information flow. All resources at the disposal of the firm have to be put to optimum use to reach these goals. All existing development processes have to be improved, and optimized to enhance productivity.

This thesis concentrates on automating the process of design in a product development environment. While the individual recipes for success vary by product type and the specific industry involved, the basic ideas remain the same for many engineering artifacts. For the purposes of this thesis, we will concentrate on the electrical motors and generators industry. We will develop a design automation tool for designing motors which can reduce the development lead time, and can lead to improved decision making during the development process.

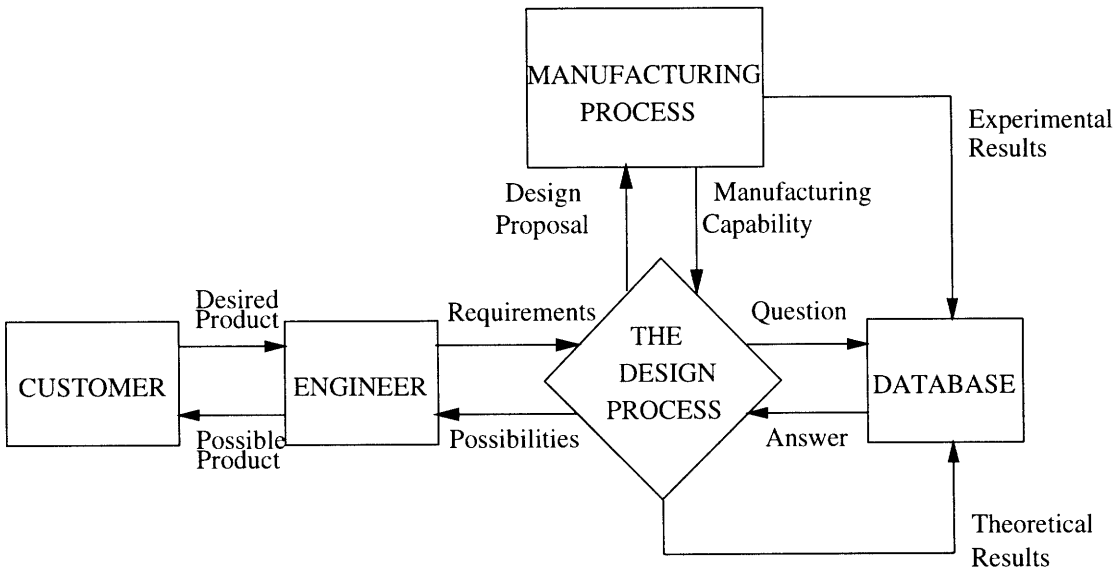


Figure 1-1: The Development Process

1.1.1 The Development Process

Broadly speaking, manufacturing processes may be classified into three categories: mass production, batch manufacturing, and job-shop manufacturing [95]. Mass production is typically characterized by large product volumes, steady (high) demand, low product variation, and expensive assembly lines. Batch Manufacturing handles medium product volumes, with volatile demand, and high product variation. Job-shop manufacturing is typically for extremely low product volumes, expensive products (e.g. a ship), and extremely high product variation. Most electrical motors (especially smaller ones – used in appliances), are mass produced, with relatively large product volumes and expensive assembly lines. However, motor manufacturing in general, also retains features of batch manufacturing, like high product variation and differentiation, and changing demand patterns.

The development process for such an industry may be viewed as shown in Figure 1-1 [72]. Moving from left to right in the figure, the customer comes in to the firm with a set of requirements and specifications. He meets with the engineer who is his contact point in the firm. The engineer then examines the feasibility of the project. He takes the requirements

through a "design process". The first step is to query the database which may be anything from a group of designers in the design division, with expert knowledge and a lot of experience, to an electronic database of all data. Using this information, the engineer comes up with a few designs which might meet the customer's specifications. During this process, frequent consultations might also be required with the manufacturing process engineers. The engineer then returns to the customer with a list of possibilities. The customer reviews the specifications and may suggest changes or enhancements. If changes are required, the design process goes through another iteration. Finally, the customer chooses a design that he is satisfied with, and places an order for manufacturing the product. Sometimes, the design process would go through many iterations before an order is placed. The subsequent series of events is straightforward (not shown in the figure). The specifications are sent to the manufacturing unit and the product is made and delivered to the customer.

1.1.2 Automating the Development Process

The entire "design process" shown in Figure 1-1 usually takes a few weeks from start to finish. Clearly, the design process is one of the important processes contributing to the development lead time. If it were possible to automate this process while retaining all the earlier benefits and flexibility, the lead time would be reduced tremendously [96]. There is evidence that at least one firm has benefited from this approach [33].

There are a number of potential areas where automation would speed-up the design process:

- Database: The design process involves frequent consultations with the database. This process can be made faster by consolidating the firm's experience into an electronic database. The designers have some expert knowledge which is very valuable to the firm, and which might be lost when employee turnover occurs. Storing this knowledge electronically, and making it accessible for all designers, would organize the learning process over the course of many development programs. Keeping all design data electronically also has innumerable advantages, especially if computer programs can access and query this database.

- Models: Building physical and manufacturing realities into mathematical models which can be evaluated using a computer would also enhance the efficacy of the design process. In the motor industry, such mathematical models typically take the form of circuit models for evaluating motor design characteristics and performance. Other possible models could include heat-transfer models, simulations of processes, models for manufacturing tolerances etc.

- The Design Process: Besides the electronic database and computer models, it is the design process itself which requires many iterative procedures, and relies a lot on the designer's expertise and intuition. Coupled with an electronic database and computer models, automating this process would have many potential advantages. Since computers can be used to perform this task, the most obvious gain is the speed at which the process can be completed. In addition, multiple alternatives can now be evaluated, which would improve the quality of the design process, often with favorable cost impacts. Various optimizations would also follow as a logical extension to this scheme.

- Information Flow: Facilitating better information flow between the parties involved is very critical for improving process efficiency. Many times, all the information might not be available at one physical location. The advantage of electronic communication cannot be over–emphasized in this context.

1.1.3 Summary of Goals for the Project

The goal of this project is to develop a computer based tool for automating this highly interactive and time-consuming process of design. The methodology used for this tool would be fairly generalizable for other industries and engineering artifacts. Specifically, this tool would attempt to accomplish the following objectives:

- React fast to customer requests. Once a customer comes in with a set of requirements, the designer should be able to respond to the customer's requests faster.

- Improve the development lead time. Reduce the total time it takes to converge on the best design possible and send it over for manufacturing.
- Facilitate multiple scenario analysis. Evaluating multiple scenarios would help improve the quality of decision making, with possible cost impacts. This in turn would add value for the customer. One of the ideas implicit in multiple scenario analysis is that the options or scenarios considered should be fairly "new" or "novel" for maximum customer impact.
- Offer better optimization capabilities. This would help with decision making and would also help understand the trade-offs involved in meeting customer demands.

1.2 Project Organization and Background

This project has been named as The Novice Design Assistant Project and the resulting system is called the NDA (Novice Design Assistant). The term "novice" is chosen since the system starts like a true novice with limited knowledge and ends up with new designs and some knowledge about the design parameters (this will be discussed in detail in later chapters).

The project was initially started as an intellectual exercise in building design assistants and over the years has resulted in a viable package which, it is hoped, would be useful to the industry.

1.2.1 Project Background

This project was started in 1989, as a collaborative effort of Professors Lang, Tabors, and Kirtley, at the Laboratory for Electromagnetic and Electronic Systems, MIT. The initial work was supported by the Leaders for Manufacturing Program. Professor James L. Kirtley Jr. wrote an analysis module for analysis of an induction motor. James A. Moses worked on the NDA and built a design assistant with a prototype database together with its database management routines and a preliminary synthesis module. This work is reported in his Master's thesis [72] and in a conference article [73]. Studies on a manufacturing simulator

and a costing module were done by Christopher L. Tucci and are described in his Master's thesis [111] and in a conference article [112].

The design synthesis ideas of the NDA were developed by the author and are reported in his Master's Thesis [96] and a conference article [97].

This thesis picks up on the previous work and enhances the features and capabilities of the NDA. The design synthesis concepts are formalized and tested with industrial strength examples. The design synthesis modules are augmented with appropriate multi-objective optimization ideas. All proposed ideas are tested on industrial examples and with different kinds of electric machines and (circuit equation) analysis routines.

1.2.2 Objective

The objective of the NDA project is to develop a software design tool which meets the goals outlined in Section 1.1.3. Such a design tool could be schematically represented as shown in Figure 1-2. The user would specify a set of performance requirements e.g. Efficiency, Power Factor etc., together with a set of (optional) constraints e.g. Stack Length not to exceed (say) 10cm. The system would then process this information and generate a set of designs which would meet the specified set of performance requirements.

For the purposes of this thesis, we define two terms to characterize the two kinds of inputs required from the user. *Design Parameters* and *Performance Parameters*. *Design Parameters* are the variables used for building up the motor. Examples are air gap, number of turns, and slot width. The customer typically is not interested in the exact values of these variables directly. However, he/she may frequently specify bounds on the values of these variables as constraints e.g. Stack Length not to exceed (say) 10cm. *Performance Parameters*, also referred to as *Attributes*, characterize a motor's performance. Examples include efficiency, power factor, breakdown torque, core volume or mass. The customer is usually interested in evaluating and characterizing a motor design based on these parameters, and specifies them as "Requirements". In addition, these attributes always have to be maximized or minimized for an ideal design. These variables are typically computed using electrical models, and most or all of the design parameters are needed for their computation.

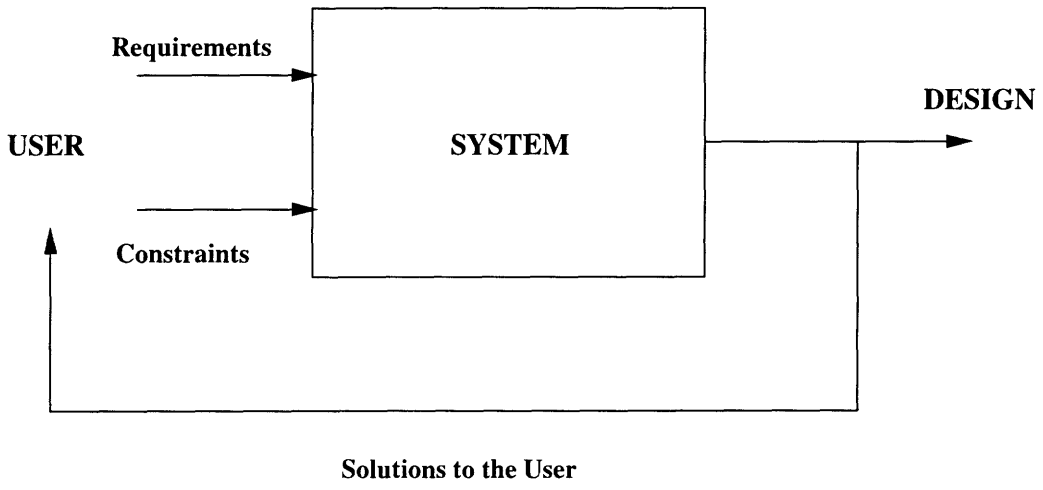


Figure 1-2: Software Design Tool for Induction Motors

1.2.3 Salient Features of Software System

The software system described above attempts to automate the design process and its various interactions as described in Section 1.1.2. The salient features of the system include:

- **Automation of the Design Process.** The user can specify requirements and constraints, and the NDA goes through a set of design procedures, and generates a list of designs which meet those demands. (In case, some of the requirements are infeasible or difficult to attain, the NDA relaxes some attributes suitably. This will be discussed in Chapter 3).

- **Optimization.** The user can then review the designs and choose to optimize based on any one attribute. The other attributes are still retained within the limits specified earlier. e.g. After reviewing the designs generated, the user can choose to optimize for efficiency while maintaining all his earlier requirements and constraints.

- **Mathematical Models.** In order to evaluate any given motor design, the system requires the necessary mathematical models. These models, which will be different for every kind of machine, need to be integrated into the NDA system. These models

are referred to as solvers in the rest of this thesis. The completed NDA has four kinds of solvers: PolyPhase solver for evaluating polyphase induction motors, SinglePhase solver for single phase induction motors, Generator solver for generators, and a 3Phase solver for simple 3 phase squirrel cage induction motors. These will be described in Chapter 5. Their sources are discussed below in Section 1.2.4.

- **Electronic Database**. The NDA communicates with a state-of-the-art electronic database which contains production design data for motors and their supporting parts e.g. slot geometries, laminations, end rings etc. The solvers communicate with the database on a regular basis to retrieve part specifications. In addition, the user can also interact with the database to store, retrieve, or view any information.

1.2.4 Relationship with Industry

For the past 5 years, the project has been supported by MagneTek Inc., and the NDA has been developed in partnership with the Advanced Development Center (Motors and Generators Division) of MagneTek in St. Louis. From 1989-1993, when the project was supported by other agencies, the NDA was designed as a fairly general system. During the association with MagneTek, the NDA was customized and specialized to work in their design environment. Consequently, the NDA is now a powerful tool for use in an industrial setting. However, it is still fairly general and extensible. Adding more solvers for different kinds of machines is fairly straightforward. The methodology, of course, remains extremely general and should be applicable to a large number of engineering artifacts.

For success in MagneTek's design environment, the NDA had to integrate MagneTek's solvers and communicate with MagneTek's database. Three out of the four solvers (PolyPhase, SinglePhase and Generator) were supplied by MagneTek. These solvers have been used extensively by the company for evaluating designs. The same solvers are accessed by the NDA to help analyze designs, and compute performance. The fourth solver (3 Phase solver) is a relatively simple solver developed at MIT. It is simply another version of the original evaluator written by Professor Kirtley in 1989.

Besides the industrial strength solvers and state-of-the-art database, the NDA also cus-

tomizes the NDA user interface and output formats for use within MagneTek.

For proprietary reasons, information pertaining to MagneTek's solvers and database cannot be presented in this thesis. Hence, all examples presented in Chapter 6 and elsewhere present only an overview picture and relevance of the results. No actual designs are discussed or presented.

1.2.5 Issues Addressed in this Thesis

Section 1.2.3 presented four salient features of the NDA system. Two of these (mathematical models and database) are provided to us by MagneTek. The other two aspects, design process automation and optimization form the crux of this thesis. These are described in Chapter 3 and Chapter 4 respectively.

This thesis is devoted to developing a software tool for designing motors as outlined above in Section 1.2.2. The tool has been developed in its entirety including the methodology and an industrial strength software implementation. The tool endeavors to meet all the goals described in Section 1.1.3.

1.2.6 Case Study Implementation

The methodology developed in this project is intended to be very general and suitable for a wide class of problems. The aim of the project is to develop a design methodology which can be used for a variety of applications. Induction motors and generators were chosen as a case study for this project since these machines are examples of mature, well–understood technology [72]. There is evidence [33] that at least one firm has been successful in this market by the rapid-turnaround of motor designs to fulfill their customer's special needs.

For the general methodology to be applicable to other engineering artifacts, two requirements need to be met from the problem:

1. All variables of interest should be clearly defined with their minimum and maximum limits, and step-sizes if applicable. These variables should be clearly divided into design parameters and attributes.

2. An evaluator or analysis routine must exist to compute the performance parameters given a complete set of design variables. Another practical consideration relates to the running time for the evaluator. For our methodology, the evaluator is invoked a very large number of times. Consequently, FEM based evaluators may not be suitable for our methodology given present day computer speeds.

Once these assumptions are satisfied, the methodology presented in this thesis should be applicable to other areas and examples.

1.3 Literature Review and Critical Analysis

Since the issues addressed in this thesis are very diverse, each chapter, as it deals with a particular topic, presents more of the literature survey and deals with some of them in greater depth. Here, we will be discussing the literature which deals with the general methodology and the philosophy of design assistants. A lot of the literature review may also be found in Chapter 2.

Various schemes have been suggested in literature for the automation of design processes. But the usual drawback with almost all of these is that these systems are very weak when it comes to synthesizing new designs.

Another important aspect tackled in this thesis is optimization. There is truly no dearth of literature related to optimization. However, there are very few optimization techniques which handle complex cases like electric machine design (refer Chapter 2).

Articles found in literature relating to design and design assistants may be classified into three categories: design optimization, expert systems, and other techniques. [6] gives a much broader overview of the methodologies used in mechanical design automation. For our purposes here, it will suffice to talk about the three categories mentioned above. We will also briefly discuss some literature pertaining to electric machines.

[11] refers to an optimization procedure which has been successfully applied to conventional optimization problems. [20] and [21] describe an optimization procedure that is based on searching the entire design space. This will be examined in some detail in Chapter 2.

Another approach in design optimization is to evaluate and redesign rather than create a design from first principles [51]. The idea is to modify a design till it conforms to all requirements and constraints. Another such example is found in [16] which employs the same technique but uses expert systems for this purpose. Moses [72] also cites a number of references which deal with non–linear optimization as a means to optimize the design of induction motors. The most general problem with optimization is that optimizing can be costly and deals with only one objective [28]. Optimizing for more than one objective at a time, is often impossible. Moreover, complex, good real designs seldom optimize a single objective; rather, they trade off performance among their various objectives. Single valued optimization does not offer much flexibility to respond to real customer preferences. Techniques for multi-objective optimization will be detailed in Chapter 4.

Recently, two techniques have emerged from the Artificial Intelligence community which offer global optimization capabilities. These are: Simulated Annealing [113] and Genetic Algorithms [47]. Simulated Annealing is based on the concept of annealing of solids and is based on the Metropolis algorithm [71]. Kirkpatrick et al [57] modified the algorithm to solve combinatorial optimization problems. Extensions to continuous variables followed immediately [114]. Genetic algorithms represent ideas inspired by natural evolution [120]. A lot of literature may be found which deals with these approaches and their applications. [22] is one such collection of papers. Both of these methods, however, require experience in fine tuning their parameters. [118] discusses an excellent design methodology which uses Genetic Algorithms for optimization. Some additional literature related to these topics is discussed in Chapter 2.

Expert Systems are used extensively for all kinds of design and synthesis work. But they lack the real flavor of innovative synthesis in them. However, they constitute a bulk of the research work in computer assisted design. [87] gives a very good outline on the use of expert systems in engineering design. A selective bibliography is also given. [83], [25], [68], [105] and [32] are excellent examples of research done in the area of Expert Systems in design. [25] also deals with optimally searching the space but uses a lot of heuristic knowledge about the problem. [83] tackles issues in interfacing expert systems with design

databases. [105] introduces expert systems which constrain design. There are two problems with using expert systems for synthesis. The traditional expert system attempts to mimic the designer's conception of the problem, building the designer's prejudices into the software. Secondly, the expert system only deals with a part of the design space that has already been explored and information about it stored as a part of the domain knowledge embodied in the expert system. There is very little scope of "creativity" per se in design. [9] and [63] outline some of the attempts to tackle this using fuzzy logic. [62] gives some background about fuzzy logic.

Expert Systems for architectural applications form an interesting class of examples. [54] models architectural design as a search process in a space of alternative solutions. [80] presents an approach to generative expert systems for architectural detailing based on design grammars and knowledge engineering. [85] is another interesting application: using design codes as expert systems. [86] and [78] are later articles from the same research group as above and very clearly distinguish between the concepts of design analysis and design synthesis. They view design as a goal oriented activity and use expert systems for both analysis and synthesis. Such an integrated approach requires the implementation of both generative and evaluative rules at different design levels within the same knowledge base. This is a powerful procedure but suffers from the same problems of lack of real "creativity". It should be pointed out however, that this approach is much more useful for architectural applications where rigorous quantitative reasoning and analysis are not encountered very frequently.

Artificial Intelligence (AI) techniques try to tackle this problem also. A lot of AI research is related to conceptual geometric design of mechanical elements. [27] deals with a design invention system which considers three general strategies for creating novel devices: generalization, analogy and mutation, all of which rely on memory organization, indexing and retrieval. [6] outlines another example which applies analogical problem solving to mechanical design. The constraint model for designing is used in [99] and [28]. [99] generates a solution tree of alternatives by processing through the static knowledge hierarchy and synthesis constraints are used to prune the solution tree. [46] and [26] give useful bibli-

ographies of books and articles relating to artificial intelligence and expert systems in the area of design. Most of the techniques outlined here help design simple systems which are categorized qualitatively. For our applications, we already have the qualitative knowledge about the design of induction motors. We are more interested in new designs which have better performance. Our demands are much more specific.

Another interesting approach to design is presented by Nam P. Suh [101], [102], [103], [104]. This approach is called Axiomatic Design. The key concepts of axiomatic design are: "the existence of domains, the characteristic vectors within the domains that can be decomposed into hierarchies through zigzagging between the domains, and the design axioms (i.e. the Independence Axiom and the Information Axiom). Based on the two design axioms, corollaries and theorems can be stated or derived for simple systems, large systems, and organizations" [102].

Specifically, there has been very little published work on computer based design tools for electric motors [72]. This, in fact, is a part of our motivation for the NDA project. One book [116] deals with this topic in some depth. It proposes several synthesis procedures which are essentially rule based, and it describes a computerized sales order specification system used in industry. This system concentrates on providing solutions to customer requests, but has limited synthesis possibilities. [61] outlines an Expert System for the Design of 3–Phase squirrel cage induction motors. Moses et al. [73] refer to other computer design tools described in [60] and [17]. [3] describes a general design synthesis and optimization procedure based on the Monte Carlo paradigm.

1.4 Thesis Overview

Before we begin a detailed description of the methodologies used and our implementation, let us outline the salient features of the thesis and present a sample result.

In order to develop the "system" depicted in Figure 1-2, we have developed a two step methodology.

In the first step, we generate 200 designs which match the user's requirements. The motivating idea behind this step is to identify a "good" region in design space which is most

likely to contain designs matching the user's given requirements. We use a Monte Carlo based design technique which uses means and standard deviations of gaussian populations to steer the random number generation process.

In the second step of our methodology, we optimize designs based on any specified design parameter while keeping all other requirements within user specified limits. This is achieved using a non-linear programming optimization algorithm. Since our space is not well characterized, we use a function approximation tool called MARS (Multivariate Adaptive Regression Splines) to obtain a surface map of the functional relationship between an attribute (requirement) and all the design (explanatory) variables. This surface map helps us compute the function values and gradients at all locations in the multi-dimensional space (and not just the 200 points generated in the first step).

These ideas are implemented in a software system which functions with four different circuit equation solvers. The four solvers are for polyphase motors, singlephase motors, generators, and a simple solver for a 3phase motor. The first three are industrial solvers and have been supplied by our corporate sponsors. The last solver is an academic example which has been developed by Professor Kirtley at MIT. Our software system has an X-Motif user interface. The entire system runs on HP9000 workstations.

We have tested our system against a number of real life industrial examples and this set of results has been very promising. We present one such experiment here, to show an example of our results. This sample run is for a 3Hp, 4 pole, 460V, 60 Hz, 3 phase induction motor. An existing industrial design with these specifications has an efficiency of 85.9%. We choose to modify and improve this design using our system. A total of three runs were performed with identical target requirements. The constraints were progressively relaxed for these three runs. In the first run (A), we allow the slot geometries and winding parameters to vary. In the second run (B), we allow the end ring to vary in addition to the variables in A. In the final run, we allow some critical geometric parameters like stator inner diameter and rotor outer diameter to vary in addition to all variables in B. A summary of our results is presented in Table 1.1. The "BaseEff" column reports the efficiency of the existing industrial design. The "Best Generated" column presents the best efficiency result from the first step

Table 1.1: 3Hp 4Pole 460V Example

Run ID	BaseEff (%)	Best Generated	Best Optimized	AveImpr (%)
A	85.9%	87.6%	87.8%	0.5%
B	85.9%	88.0%	88.0%	0.5%
C	85.9%	87.8%	88.0%	0.5%

of our methodology. The "Best Optimized" column presents the best efficiency design from the second step of our methodology. The average efficiency improvement between the first and second steps is shown in the "AveImpr" column.

The improvements suggested in Table 1.1 are significant. We will present additional examples, results, and analyses in Chapter 6.

We hope the NDA would be a viable design tool in industry and we look forward to gathering a lot of collective industrial experience with our methodologies and software system.

1.5 Thesis Organization and Road Map

Every chapter addresses a specific topic and begins with a brief description of the previous work done on that topic and in that field in general. Problem formulation and related literature survey are tackled in Chapter 2. The issues tackled in this thesis have been divided into four modules for the purposes of the NDA. The first module is presented in Chapter 3, and deals with identifying a relevant region in the design space which is likely to meet the user's requirements. This is the primary module for automating the design process. The second module deals with optimization. These issues are detailed in Chapter 4. The third module deals with the aspects of integrating different solvers in the NDA. The fourth module was the user interface for the NDA. The third and fourth modules are presented in Chapter 5. Chapter 6 presents the results obtained by running the NDA, and analyzes

those results. The thesis concludes with Chapter 7 which summarizes the thesis and presents directions for future research.

Chapter 2

Problem Formulation, Alternative Approaches and Proposed Methodology

As outlined in Section 1.2.2, the objective is to design a software design tool where the user can specify Requirements and Constraints and completed designs are returned to the user. In this chapter, we discuss how we break down this rather abstract problem into manageable pieces. Section 2.1 deals with the problem formulation i.e. how we think about the problem and the framework we lay down for solving it. In Section 2.2 and Section 2.3, we present some techniques for solving the presented problem. Section 2.4 outlines our proposed methodology which will be examined in detail in the following chapters.

2.1 Problem Formulation

2.1.1 Modules of the NDA

As mentioned in Section 1.2.3, the NDA has four parts together with a user interface. The user submits requirements and constraints using the User Interface. This information is then sent to the Design Automation (or Design Synthesis) module. The Design Synthesis

module generates a few designs while using information from the Database and Circuit Models (Solvers). After the design generation phase, the generated designs are sent back to the user, who may decide to proceed with optimization. The Optimization module uses information from the Database and Circuit Models, and the Design Synthesis module. The optimized design is again sent back to the user interface. In this simple operation process of the NDA, Design Synthesis necessarily precedes Optimization and the user interface ensures that things happen in a logical sequence. The Modules of the NDA are schematically shown in Figure 2-1.

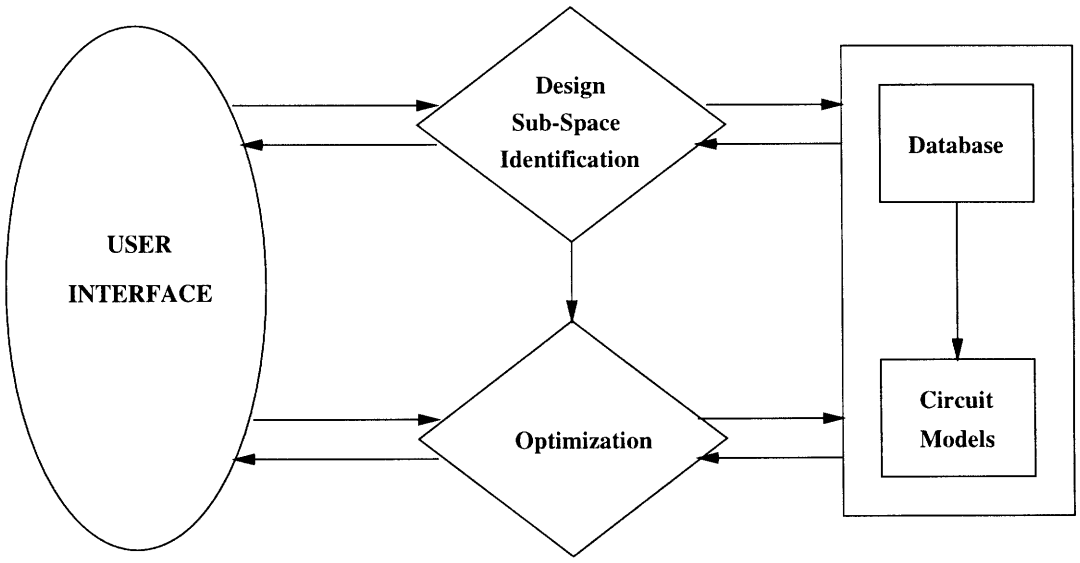


Figure 2-1: The Different Modules of the NDA

In the following Sections, we will discuss how we formulate the problem of Design Synthesis and how the design process is modeled in the Design Synthesis module. The Optimization module follows from pretty much the same problem formulation, but differs in content and function. The Optimization module is described in Chapter 4.

2.1.2 The Design Process

"Design" may be considered to be of three types or levels [86]. The first and most difficult is the design of an artifact to satisfy a set of goals when even the general form of the artifact is not known. Let us say we do not know that a household washing machine exists. The top level of design would deal with the problem of inventing a machine or a concept which would be useful for washing clothes conveniently. The second level of design deals with the problem of deciding design parameters when the general form of the artifact is known. This is one of the common "hard problems" in engineering. In the washing machine example, this problem would correspond to a scenario where we know the general form and function of a regular washing machine, and the problem is to design a machine which would meet our specific requirements. At the bottom level of design is the process of classification and selecting from a set of fully or partially described solutions. This is similar to selecting the best washing machine available in the market, or from a catalog.

The first level design problems are tackled by some artificial intelligence techniques. References for some AI techniques have been presented in Section 1.3. We do not deal with this class of problems in this thesis.

The third level of design was tackled in a previous work on this project [96], where a database was designed to facilitate easy lookup and storage of motor designs. Another common example of the third level of design is bearing selection [32]. This level of design is also not specifically addressed in this thesis. The concepts have been demonstrated earlier [96], and for the purposes of this project, database lookup and other database interactions are accomplished using in-house software tools and facilities developed by our industrial sponsors.

In this thesis, we focus only on the second level of design. Typically, in such problems, the issue is to generate a design which would match certain performance requirements.

2.1.3 Design Synthesis

Design Synthesis (in our context) is the process of generating a new design starting from certain performance requirements. This is in contrast to the process of analysis where

performance is calculated starting from a given design. Usually in engineering practice, and indeed in the area of Electric Machine design, the analysis process is much better understood. The performance calculation equations, and all interrelationships are well established. The process of Synthesis, on the other hand, is much harder. It is also the more common operation in industry. Traditionally, synthesis has been attempted by designers using their insight and experience. This experience is in the form of certain non–formalized "thumb rules" and guidelines. Designers develop intuition about the logical outcome of making certain changes to some of the design variables. This accumulated experience and intuition helps the designers in their designing process, which in essence, is a systematic and organized trial and error process.

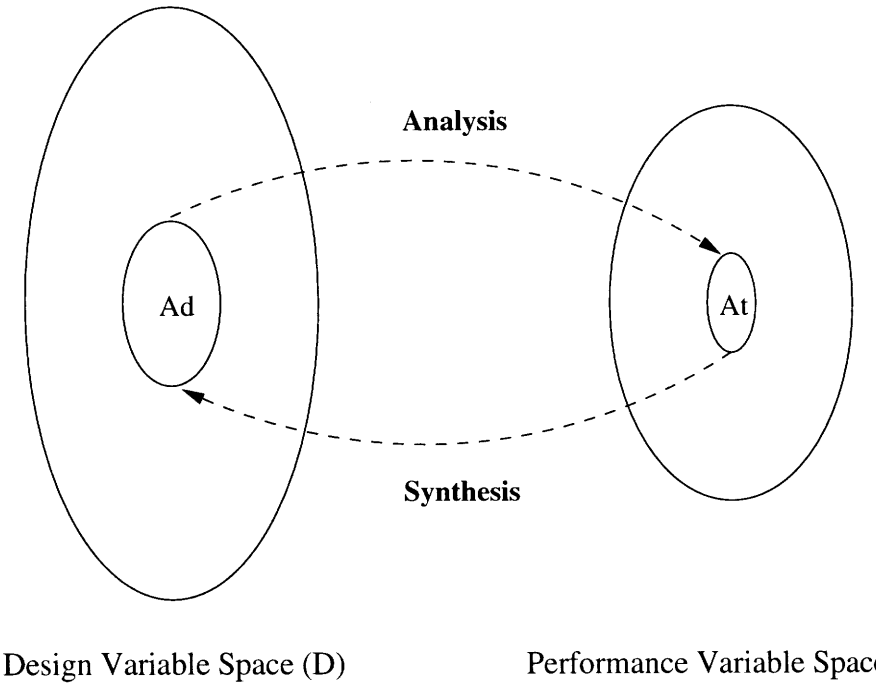


Figure 2-2: The Design Process - Variables in Design and Performance Space

This difficulty in synthesizing designs can be better explained using the concept of *mapping*. Let us define a design variable space **D** representing the possible values that each of the design variables can assume. Similarly, **P** is the performance variable space

representing the possible values of the performance variables. The process of Analysis is then defined as a mapping from the design variable space to the performance space. This is the forward map. Synthesis is the backward map or an inverse mapping from the performance space back to the design variable space. Given an acceptable target region in the system characteristics space, A_t, the problem is to find a corresponding acceptable region A_d in the design variable space. This is shown in Figure 2-2 [81]. It is this backward map which is difficult to achieve in a straightforward manner, primarily owing to non–linearities in the mapping.

Let us denote our set of design variables by an n $\times$ 1 vector.

$$\mathbf{x} = [x_1, ...x_n]^T \quad \text{where} \quad x \in \mathbf{D} \tag{2.1}$$

and our set of performance variables by an m $\times$ 1 vector.

$$\mathbf{y} = [y_1, ...y_n]^T \quad \text{where} \quad y \in \mathbf{P} \tag{2.2}$$

The forward map from $\mathbf{D}$ to $\mathbf{P}$ is then defined as a function from $\mathbf{D}$ to $\mathbf{P}$.

$$\mathbf{y} = f(\mathbf{x}) \tag{2.3}$$

and the backward map is defined as the inverse function.

$$\mathbf{x} = f^{-1}(\mathbf{y}) \quad \text{or} \quad \mathbf{x} = g(\mathbf{y}) \tag{2.4}$$

2.1.4 Important Issues Involved

For our problem of electric machine design, it is this inverse map $\mathbf{g}$ or $\mathbf{f^{-1}}$ which is nearly impossible to obtain. Equation 2.3 is so involved that it is extremely difficult to write it explicitly. Writing Equation 2.4 is out of question. We can identify six primary reasons why the process of synthesis is so hard for our problem:

- **Non–linearities**. The mapping **f** is highly non–linear. Many techniques exist for manipulating linear equations, but there are relatively few techniques for managing non–linearities elegantly.

- **Coupled Equations**. The equations in Equation 2.3 are highly coupled. In most solvers, some iterative procedures are used to arrive at certain intermediate quantities (e.g. slip in an induction motor).

- **Discrete and Categorical Variables**. Some variables x_i of Eq 2.1 are discrete in nature. Examples include Number of Turns of Wire in a slot. There are some others which are purely categorical (i.e. cannot be ordered) e.g. material for stator and rotor iron. Most mathematical techniques handle only continuous variables. The few techniques which handle categorical variables are not as powerful or general.

- **Dimensionality**. We are dealing with a large number of variables which adds to the complexity of the problem. The number of possible combinations of these variables grows exponentially with the number of variables.

- **Multiple Objectives**. Our performance variable space is multi-dimensional i.e. we deal with a number of attributes concurrently. For example, for a polyphase design, we are interested in Efficiency, Power Factor, Breakdown Torque, Locked Rotor Torque, Locked Rotor Current, Slot Fill, and Core Volume. Most mathematical techniques (especially optimization techniques and non-linear techniques) deal with only a single attribute.

- **Under-Constrained Problem**. The dimensionality of the design space is much higher than the dimensionality of the performance space.

Another important issue which merits attention at this point is the relative speed of the design analysis and synthesis operations. Usually, analysis processes are better understood and formulated, for most engineering artifacts. For example, for electric motors, we have circuit models which help us compute the performance of a given design. Consequently, this process is relatively faster. Synthesis, on the other hand, is not so conveniently performed.

Many synthesis techniques apply analysis routines multiple times to perform the synthesis operation. As a result, synthesis is usually slower. This has important implications especially for cases where the analysis is not fast e.g. FEM models. Synthesis in such scenarios is even harder, much slower, and perhaps less accurate. For the purposes of this thesis, we will not consider such cases since our analysis routines take on the order of 5-10 seconds to run. However, in Section 4.2 we will introduce a function approximation technique which can be used to circumvent this problem, should it arise in other research contexts.

2.2 Alternative Approaches found in Literature

The problem formulated in Section 2.1 is a common problem encountered in engineering analysis and different researchers have used different simplifying assumptions and techniques to circumvent its inherent difficulties. Most of the ideas described below may be found in the author's Master's thesis [96]. Some related work has also been presented in Section 1.3. Consequently, the discussion here will be brief. We mention these techniques here for the sake of completeness.

2.2.1 Expert Systems

Expert Systems embody a core of knowledge about a (very) specific field. This knowledge is compiled by a body of experts and captures the "experience" of the experts. Usually, knowledge in expert systems takes the form of *"if ... then ..."* rules. Hence expert systems are ideally suited for the third level of design. With this class of problems, the goal driven or backward chaining process of Expert Systems can be used to accomplish all design goals. Expert Systems have also been used for the second level of design by trying previously known solutions. Good discussions may be found in [78] and [86].

Expert Systems are unsuitable for the NDA since they lack the flavor of real "creativity" in synthesizing solutions. The NDA is intended to be useful for designers who are looking for novel and innovative designs, usually something they have not encountered before.

2.2.2 Grid Search

Another popular technique is to search all possible solution states i.e. search the entire (n–dimensional) grid of solution states in the design variable space. Usually, as the number of design variables gets large, this brute force approach becomes computationally prohibitive. It is useful and satisfactory for simple problems where the number of design variables is small and the number of grid–points (on every design axis) is fairly small. Nevertheless, techniques based on the grid search have been very popular. Most of them attempt to avoid searching the entire grid. They search only a part of the grid and arrive at an acceptable solution. There is an interesting application where concepts of Knowledge Engineering and Expert Systems are combined with grid search to develop a powerful (application specific) design methodology for design optimization [20] and [21]. Grid search ideas have also been used for optimization [11].

Our solution space is extremely large (has high dimensionality) and this technique is therefore ruled out. The NDA problem does not satisfy other requirements of [20] either which eliminates the possibility of combining grid search with knowledge engineering.

2.2.3 Linearizing the Mapping

Many researchers have approached the above defined problem by attempting to linearize the mapping about a nominal design point. Rai [81] has proposed a Singular Value Decomposition based approach which has been very successful with robot design examples. This method assumes that the designer has a nominal design as the starting point. This starting design meets the basic requirements of the designer. Starting from this nominal design point, the space in close proximity of this point is linearized, and preferred directions for maximizing the sub–goal in performance (for that step) are computed.

This approach has the attractive property that multiple objectives can be very conveniently handled. However, it is still unsuitable for the NDA since our space is very highly non–linear. A lot of information will be lost in the linearization process and may lead to erroneous results. We also want to avoid specifying a feasible design a priori, which would sacrifice the "creative aspect" in design synthesis.

2.2.4 Simulated Annealing

Simulated Annealing is based on the concept of annealing of solids and is based on the Metropolis algorithm [71]. Kirkpatrick et al [57] modified the algorithm to solve combinatorial optimization problems. They were the first to solve the famous Traveling Salesman problem using this technique. Extensions to continuous variables followed immediately [114].

Consider a problem of minimizing a function $E(x) = E(x_1, x_2, ..., x_n)$ defined over an n-dimensional continuous parameter space. The most basic Simulated Annealing algorithm proceeds by choosing an initial starting point x_0 and making random steps Δx. At each step, the change in the objective function ΔE is evaluated. If ΔE is negative, the step is accepted. If ΔE is positive, the step is accepted with probability

$$p = exp[-\Delta E/T] \tag{2.5}$$

The series of accepted steps then generates a random walk which explores the parameter space. The parameter T plays the role of temperature: as T is decreased slowly, the system is forced to "freeze" or "anneal" into the configuration of lowest E.

This method is very useful for navigating spaces spaces with many local optima. It is also very good at handling non–linearities, discrete variables and coupled equations. However, there are a number of potential problems with this simple scheme. For example, deciding the cooling schedule is a complex and non-trivial task. Usually, some experience with the problem is required for cooling schedule selection. [88] also points out that the performance is poor in optimization problems with many inequality constraints. [88] proposes an improvement to the algorithm to overcome this shortcoming. For the purposes of the NDA, this technique is unsuitable for a variety of reasons. First, it is extremely hard to handle multiple objectives with this method. Second, inequality constraints are an important part of our problem formulation. Third, this technique requires a lot of knowledge or experience about the problem to decide cooling schedules.

This algorithm has received a lot of attention from the research community and conse-

quently there is a lot of literature available on this topic. [113] is a good text which describes simulated annealing in detail. [22] is a good collection of papers on Simulated Annealing and Genetic Algorithms (which have similar capabilities). [88] presents an application to nuclear power plant design. [12], [110], [44], [45], [106] and [107] present a sampling of some work on the theoretical aspects of this algorithm.

2.2.5 Genetic Algorithms

Genetic Algorithms represent a technique inspired by natural evolution. They rest on ideas that are analogous, in some ways, to *individuals, mating, chromosome crossover, gene mutation, fitness,* and *natural selection.* [120] provides a good introduction, whereas [47] is perhaps the bible on this topic.

The example chosen in [120] attempts to optimize the proportion of flour and sugar in a cookie recipe. A "chromosome" here consists of two "genes", each of which is a number from 1 to 9. The first of these specifies the amount of flour and the second specifies the amount of sugar. Each of these genes is altered randomly (keeping within limits) to produce new "individuals". This mimics the process of mutation. To mimic crossover, genes from two individuals are cut and joined across. Also, a quality score is maintained for each of the individuals which helps compare individuals.

Fitness can be quantified by a formula:

$$f_i = \frac{q_i}{\sum_j q_j} \tag{2.6}$$

where q_j is a quality score and f_i ranges from 0 to 1.

Once we have a quantitative definition of fitness, natural selection is easy to model, based on fitness scores. This process is used to determine which individuals would survive and go on to the next generation. This process of mutation, mating, crossover and selection is repeated for a number of generations. The resulting population is very rich in individuals with desired traits.

Wallace et al [118] present a specification-based design evaluation method and apply it

to optimization using Genetic Algorithms. This paper presents a way of quantifying design goals and formulating search problems using the familiar language of design specifications. This model is intended to provide designers with an intuitive method to define objective functions for multiple-criteria problems and to provide a robust design search mechanism. The idea is to present an acceptability function for every design specification. This function is the probability that a designer will accept a design with the given attribute. For example, if a particular design uses a material with a cost ratio ($\frac{\text{Material Recycling Cost}}{\text{Virgin Material Production Cost}}$) of 0.5, the subjective probability that the designer will accept the design is 1.0. Whereas, if the cost ratio is 1.5, the probability of acceptance is 0.0. The probability of acceptance may decrease linearly between cost ratios of 0.5 and 1.0.

This is a very powerful formulation for multiple attribute design (which is the usual drawback with most genetic algorithm based formulations). As long as each of the target attribute limits are independent of the values or levels of other attributes, the probability of acceptance based on multiple attributes can be readily quantified as the product of the individual probabilities. A GA based solver is used to search for designs which would match the user's target attribute limits.

The technique presented in [118] is intended to be a flexible object oriented modeling framework, where different specific pieces (like solvers and search engines) can be plugged in. The framework is extremely general and can be applied to a wide variety of problems. In many ways, it is more general than the technique proposed for the NDA in this thesis. The technique of [118] has also been shown to work with large problems with a large number of variables. Since the technique uses genetic algorithms, problems related to non–linearities, discrete variables and coupled equations are easily bypassed. It would be interesting to compare the performance of this system with the first step of the NDA algorithm which has achieves a similar objective of generating design alternatives which would match a designer's target limits.

Like Simulated Annealing, Genetic algorithms have received a lot of attention from the research community in recent years. Consequently, there is a lot of literature available on this topic. We will mention a small sample of references here. [76] presents an application

of Genetic Algorithms to motor parameter determination. [75] is a similar application to power transformer design. [22], as mentioned above, is a good collection of papers. [117] offers an excellent treatment of genetic algorithms (with implementations).

2.2.6 Monte Carlo Approach

The Monte Carlo process is very similar to Genetic Algorithms in that random numbers are used to generate the values of design variables, and the resulting design is evaluated for fitness. This approach was initially inspired by the Monte Carlo simulation approach used in statistical analysis. The basic concepts remain the same here. We generate random numbers and map them to the range of allowable values for every design variable. The resulting design is then analyzed. This approach is adopted for the purposes of the NDA and is discussed later in this chapter and in Chapter 3.

The Monte Carlo process has been outlined in [96]. [108], [109], [98], and some of the references therein are other examples of applications of the monte-carlo process. [19] is an interesting application of the monte carlo technique to optimizing a cluster of wind turbines.

2.3 Other Alternative Approaches

2.3.1 Conventional Optimization

Optimization is a very well developed field and a lot of work has been done in that area. There is truly no dearth of books and articles on constrained and unconstrained optimization. There are however, no constrained non-linear optimization techniques which guarantee a global optimum. Still, conventional optimization has been widely used for electric machine optimization and some of the articles are mentioned here. Chapter 4 will present a methodology which uses conventional optimization in combination with other concepts resulting in a very powerful algorithm.

[34] describes an optimization technique for three phase induction motors, based on the annual cost of the motor. It uses twelve design variables and seven performance indices which are used as inequality constraints. [4], [93], [82], [8], [60], and [17] describe computer

based design and optimization techniques for induction motors. [67] and [66] describe a technique for decomposing the problem of motor design optimization and solving the problem in two stages, thereby attaining a "global" optimum. All of these articles, however, deal with continuous variables only, which is a major limitation when it comes to using their methodologies in a truly industrial strength design setting. [119] discusses a finite element analysis based technique for design optimization of rotor slots.

2.3.2 Polytomous Logistic Regression

This is a relatively new statistical technique and is based on the concepts of Categorical Data Analysis. A good introduction to multivariate categorical data analysis may be found in [1]. [10] is another useful reference for discrete multivariate analysis. Polytomous Logistic Regression, being a more advanced topic is only introduced in these texts. A more thorough treatment may be found in [70]. We will not delve into the details here but would merely introduce the topic. Given our current state of technology, these methods are impractical for the NDA (where the problem size is large).

The biggest advantage of using these techniques is that these are techniques for mixed categorical and ordinal variables. Hence, all kinds of data analyses can be performed.

In the simplest sense, logistic regression tries to model the (logarithm of the) odds of the response variable to be in a certain category. For ordinal responses, we can visualize that the response is divided into a number of categories or bins. The proportional odds of the response being in a certain category are regressed against the explanatory variables. One simple form is the ordinal logistic regression model using cumulative logits [70]:

$$log\frac{\gamma_j(\mathbf{x})}{(1-\gamma_j(\mathbf{x}))} = \theta_j + \beta^T\mathbf{x} \tag{2.7}$$

where $\gamma_j(\mathbf{x}) = Pr(Y \leq j|\mathbf{x})$ is the cumulative probability up to and including the category j, $\mathbf{x}$ is the vector of explanatory variables, and β represents the regression coefficients.

When there are multiple responses (attributes), then polytomous regression is used. The governing ideas are similar but multiple responses and interactions between different responses can also be modeled.

A complex routine for performing Polytomous Logistic Regressions (based on the Splus software package) has been written by Professor Alan Zaslavsky of Harvard University. This program models interactions between responses and between the explanatory variables. We tried to use that routine to see if the ideas would be applicable to the NDA. Unfortunately, for a very large problem like ours, we ran into computational limitations. In one case, where we used only 5 design variables (out of a possible 50), the model took about 5 hours before it gave any results on a Sun SPARC Station. Increasing the number of design variables would increase the time exponentially. So, we will document this as a technique which might be useful, and may become important a few years later.

2.3.3 Neural Networks

Neural networks are inspired by biological neural systems which learn from experience. They are function approximation techniques which help us model the relationship between two sets of data. From this perspective, they are very similar to regression. However, with neural networks, we do not obtain any analytic relationship (in the form of equations etc.) between the sets of data.

An excellent introduction to neural nets may be found in [65]. [7] is one example of an application related to electric machines. Usually, a neural network (NN) is viewed as a highly parallel ensemble of computing elements or neurons [7]. These neurons are connected with other neurons in the NN. One simple framework (feed-forward strategy), neurons are organized into 3 layers: input, hidden, and output. The layers are interconnected with weights that represent the information and function stored in the NN. Input and Output layers are useful for taking in input and supplying output. The hidden layer contains a hidden representation of the system being modeled. This NN is trained using a back-propagation algorithm. This algorithm essentially takes input/output training pairs, computes the errors at the outputs, and then back propagates these errors back through the network. This helps update the interconnection weights. The transfer function of a typical

bipolar neuron is a sigmoidally shaped function which may be shown as:

$$F(net) = \frac{1 - \exp^{-net}}{1 + \exp^{-net}} \tag{2.8}$$

where *net* represents the weighted sum of inputs to the neuron and $F(.)$ is the new output from the neuron.

Neural nets help approximate functions which can be useful in many design scenarios for estimating design parameters after training the neural net with previously existing data.

[94] describes an application of neural networks to economic load dispatch. [5] presents an application to estimation of Induction Motor parameters based on FEM results. [7] has modeled Switched Reluctance Motors using a combination of Neural Networks and a type of genetic algorithm (referred to as Evolutionary Algorithm). An interesting overview of the powerful combination of fuzzy logic, neural networks and genetic algorithms may be found in [18].

2.4 Proposed Methodology

2.4.1 Monte Carlo Approach

Based on our survey of existing techniques, it appears that the Monte Carlo Approach serves our needs best. Since the approach is based on random numbers, it does not have to deal with navigating highly non–linear spaces. Points in the space are chosen "randomly". Similarly, an approach based on random numbers is free from problems posed by discrete variables and coupled equations since equations and variables are never manipulated directly. It is however, essential to use some expert knowledge to condition the problem (usually the initial bounds on the values for the design variables).

A Monte Carlo based technique is an extremely "creative" problem solver since the approach is based on random numbers. Lack of "creativity" was one of the major problems with the Expert System approach. Grid Search was computationally prohibitive and Linearization led to a loss of information. All of these problems are bypassed with the Monte Carlo approach.

The Monte Carlo approach differs from optimization of a scalar objective function in many ways. First, there is no objective function to optimize. The objectives are given in terms of user requirements which have to be met. Second, a multi–attribute design synthesis is performed using the Monte Carlo approach. No attribute weighting is required, as might be necessary in formulating a single scalar objective function. Third, the constraint equations are not manipulated directly, as in scalar optimization.

The ability to deal with multiple attributes is a distinct advantage over Genetic Algorithm and Simulated Annealing techniques which are primarily designed for single attribute optimizations.

The rest of this thesis is devoted to developing a two-step methodology using the Monte Carlo process as the basis.

2.4.2 Design Sub-Space Identification

The first step of our two step methodology is Sub-Space Identification. The idea behind this step is intuitively very simple. Let us imagine all the n design variables to span an n-dimensional design space. Every point in this space would represent a combination of variables which could (possibly) constitute a design. However, it is very clear that certain regions of the design space would be useless. For example, all motors with *extremely* long lengths and *extremely* small diameters, though permissible in this mathematical construct, do not represent viable physical motor designs. The Design Sub-Space Identification step endeavors to identify regions of the design space which are likely to contain viable designs which meet the user's requirements.

However, this task of identifying the sub-space is not straight-forward. This is due to the fact that the variables are not independent of each other, and interactions between them are important. For example, extremely large motors are highly unlikely to have really small slots. In a slightly more involved example, for the same motor with identical slot geometries, the number of turns of wire in every slot will vary depending on the wire diameter. While a large number of turns would be acceptable for thin wires, it might not be physically possible with thicker wires. Hence, it is very difficult to specify bounds for individual variables in

an attempt to identify the active sub-space.

Chapter 3 is devoted to developing design sub-space identification ideas based on multi-normal distributions of variables. The identified sub-space is defined in a probabilistic sense as opposed to defining fixed boundaries for variables.

2.4.3 Optimization

Once the active design sub-space is identified, our goal is to search for "optimal" designs in this sub-space. For this purpose, some of the strategies described above in Section 2.3 or Section 2.2 might be used. Conventional Optimization offers capabilities to deal with multiple attributes. But it is difficult to use with discrete variables and non–linearities, often resulting in local optimum solutions. Sometimes, it even leads to instabilities in the search process.

Combining with the Design Sub-Space Identification step offers an obvious solution. At the Design Sub-Space Identification step, we have a sample of designs which meet the user's specifications. These designs are used as (multiple) starting points for the optimization step. This helps deal with the discreteness and non–linearity problems associated with our space.

Another problem with conventional optimization is that it requires an objective function and its gradients. Similarly, gradients for all constraints are also required. Our objective function value is computed using circuit models. The gradients, however, are a problem. To address this issue, a technique which combines MARS (a function approximation technique) and a non–linear constrained optimization method are used in conjunction. These ideas are detailed in Chapter 4.

Optimization based on multiple attributes requires more careful attention than optimization for one attribute. These ideas are also explored in Chapter 4 and suitable strategies for multi-objective optimization are presented.

2.5 Recapitulation

In this chapter, we have formulated our problem for developing a software design tool for the NDA. A conceptual model is proposed which is very useful for decomposing the problem into manageable chunks. Techniques used in literature to tackle similar problems in related or other areas are presented. In addition, some other techniques are also presented which may conceivably be used for our kind of problem. Finally a two step methodology is proposed for solving the problem. This two-step methodology is detailed in the following two chapters.

Chapter 3

Design Sub-Space Identification

The Monte Carlo process has been used in the first step of our design methodology. In this chapter, we discuss the process of design sub-space identification. Section 3.1 outlines how the Monte Carlo process can be used for the synthesis problem formulated in Chapter 2. The general methodology for design sub-space identification is described in Section 3.2, and certain statistical concepts used to augment our algorithm are presented in Section 3.3. Implementation comments and an example are presented in Section 3.4. The chapter concludes with a brief summary.

3.1 The Monte Carlo Process

3.1.1 Using Monte Carlo for Design Synthesis

The Monte Carlo process, in a nutshell, means using randomly generated numbers. We use some domain knowledge to establish upper and lower bounds for all design variables. Randomly generated numbers are mapped onto the established ranges for each design variable. After we have generated values for all design variables, we get a randomly generated design, or a point in the $n-$dimensional design space.

But how does this help with design synthesis? Referring back to Figure 2-2, the Monte Carlo process has given us a point in the design variable space. This can be conveniently analyzed using circuit models to evaluate the performance from this given design.

This process, if performed multiple times, could be useful for design synthesis. After a randomly generated design is analyzed, it is evaluated to see if the resulting performance meets the target specifications. If the design meets the requirements, it is retained, otherwise, it is rejected. This process is repeated till a list of acceptable designs is found. This process is shown schematically in Figure 3-1.

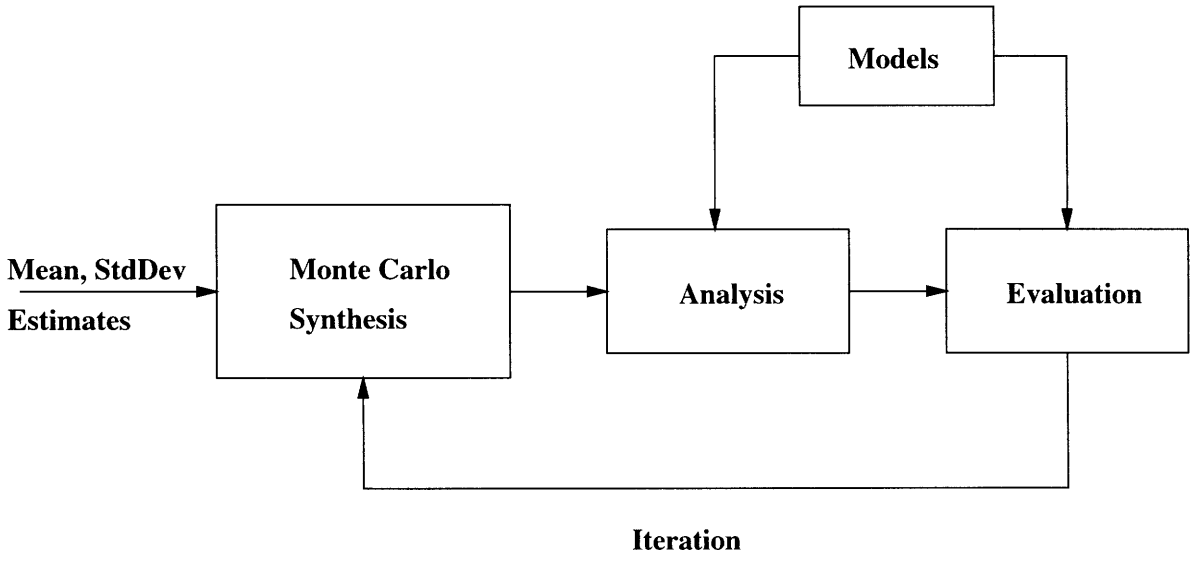


Figure 3-1: The Design Synthesis Cycle

This was the basic Monte Carlo process used in [72]. This process has been suitably modified and enhanced in [96] and in this thesis to serve our needs. We will first present the problems associated with this "pure" Monte Carlo process before we describe our algorithms which address those problems.

3.1.2 Problems with Pure Monte Carlo

The Monte Carlo Approach, though most suitable for our needs, is not free from problems. The first concern is Expert Knowledge. By using more expert knowledge, we are compromising on the "creative" element in the designs. By using little expert knowledge, we end

up with a large number of infeasible designs, and hence need to generate many more designs before we arrive at our set of successful designs.

Apart from this trade–off, pure Monte Carlo is never guided towards feasible designs and hence "convergence" to a feasible design is impossible to predict. Generation of a feasible design is a matter of chance. There is no bound on the time it might take to generate a design. On one (lucky) extreme, the first run might give a feasible design and on the other extreme, the program may run for days on end without results! This issue is addressed by our design sub-space identification algorithm.

Yet another important issue with the Monte Carlo Approach relates to conditioning of variables. For example, if geometrical parameters are allowed to vary "randomly", it is very difficult to ensure that there is no "negative geometry". Similarly, relationships between variables are harder to maintain. For instance, in a motor design, if both stator inner diameter and rotor outer diameters are allowed to vary randomly, it is very difficult to maintain a reasonable air gap between the two. Air gap typically ranges from 0.015 to 0.05 inches! There are a couple of obvious solutions to this problem. One solution might be to use a modified set of variables e.g. instead of rotor outer diameter and stator inner diameter, we use rotor outer diameter and air gap! This approach has been used with the MIT 3Phase Solver (Section 5.4.6). Another approach might be to do some sanity checks (using some very basic expert knowledge). Since the NDA is designed to be flexible and problem independent, this approach is perhaps more suitable. In the NDA, this approach has been used by having a rule sets module for every solver which performs the requisite sanity checks (Section 3.4.2).

In this chapter we will present a methodology which tackles these problems successfully. Some expert knowledge is used to address issues related to initial bounds and conditioning of the problem. Our design sub-space identification algorithm addresses the convergence issues.

3.2 Methodology for Sub-Space Identification

3.2.1 Overview

There are two motivations for the Design Sub-Space Identification algorithm. First, we want to identify a region in the design space which is likely to contain designs matching the user's requirements, and constraints. Second, we want to improve some of the convergence characteristics of the Monte Carlo Process.

At the very outset, we must emphasize that any attempt to improve the convergence characteristics of the Monte Carlo process will potentially hamper the "creativity" of the process. Most attempts to guide the Monte Carlo process would tend to steer the process in certain preferred directions. This means that certain other regions of the design space would be left unexplored. Potentially, these unexplored regions could contain some viable designs. Hence, when we use the Monte Carlo process, we are always faced with a trade-off of practicality (speed) versus creativity. Our methodology for design sub-space identification is no exception. By using the algorithm, we could potentially miss some designs. However, for the purposes of the NDA, we have tried to retain as much creativity as possible in the process. Consequently, the design sub-space algorithm runs for a relatively longer period of time as it explores a large number of possibilities.

For understanding the design sub-space identification methodology, we refer back to our problem formulation presented in Figure 2-2. The problem of synthesis is the process of finding a corresponding map A_d in the design variable space for a target region A_t in the performance variable space. Since a direct and straightforward synthesis process is difficult, we use the analysis process multiple times to achieve this goal.

3.2.2 Algorithm

The underlying idea behind the methodology is to solve an easier problem first, learn from it, and then solve a slightly harder problem. This process is repeated till we solve our given problem.

Consider Figure 3-2. We first try to solve an easier problem where the target region

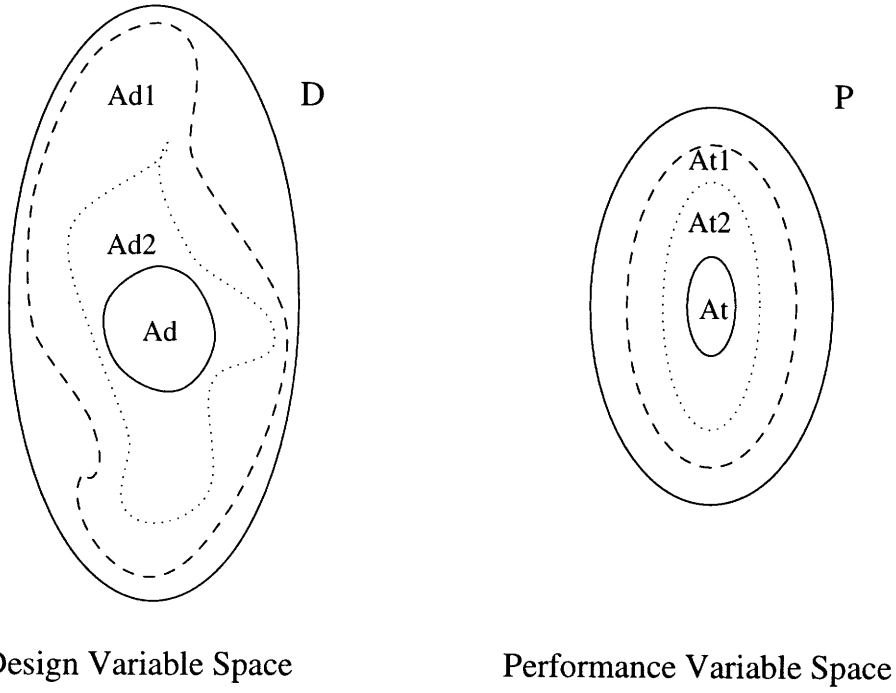


Figure 3-2: The Design Sub-Space Identification

in the performance space is A_{t1}. The Monte Carlo process is repeated multiple times to generate a good number of designs which meet the performance requirement of A_{t1}. The design variables from this set of "successful" designs are used to identify the corresponding region A_{d1}. (In Section 3.3 we show how exactly this is done using statistical tools).

Once we have an idea of the design sub-space A_{d1}, we can use this knowledge to solve a slightly harder problem. We constrain our target performance to A_{t2}, and repeat the above sequence of steps to identify A_{d2}. This entire process is repeated till we converge on our desired target performance A_t, and identify the corresponding A_d.

The methodology is also shown in Figure 3-3 in an algorithmic fashion.

We start with the global limits of the performance variable space. These limits are known from simple expert knowledge. For example, we know that efficiency of a machine would lie between 0 and 1. Such global limits are established for all performance variables to begin with. At the next step, these limits are constrained. This is done using a simple

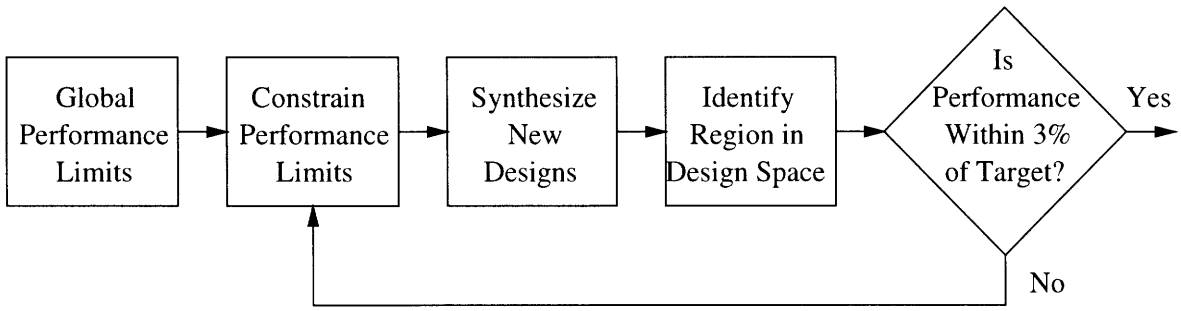


Figure 3-3: The Design Sub-Space Identification Algorithm

heuristic where we tighten bounds to 1/3 of the difference between the current bounds and the desired target performance limits. Following our example of efficiency (with global bounds of 0 and 1), let us say the user's desired target was between 0.8 and 1.0. In this step, we would fix our interim target A_{t1} to range from 0.27 and 1.0 for efficiency. Other performance variable targets are decided similarly. If we had Power Factor as another performance variable with global limits of 0 and 1, and user's target between 0.7 and 1, the interim target A_{t1} would reflect a target ranging between 0.23 and 1 for Power Factor.

Once the interim targets are decided, Monte Carlo design synthesis (Figure 3-1) is performed to identify the corresponding region A_{d1} in the design variable space. Clearly, at this stage, our target performance is not close to the user's desired limits. Hence, we constrain our target performance further and repeat the entire process till we obtain a good number of designs which meet the user's requirements.

One of the questions which comes to mind is: Why 1/3? As alluded to before, this is simply a heuristic. One could as well fix this parameter to 1/6 or any other convenient fraction. We have observed that 1/3 seems to be suited for our problem, considering the trade-off between creativity and practicality of the algorithm.

Another issue which deserves mention is that all performance variables use the same fraction (1/3). Why don't we decide different fractions for different performance variables?

A careful look at the algorithm reveals that having different fractions does not change the algorithm or its performance in any way! These fractions are really a measure of how fast we want to "converge" our performance parameters to the target limits. Let us say that we used different fractions for Efficiency and Power Factor, and had a higher fraction for Power Factor (converged it faster). After a few iterations, even though we would meet the user's Power Factor targets, the process would continue unchanged, because other performance requirements would not have been met. For the design sub-space identification process to complete, we need *all* performance variables to meet the user specified targets. Hence, there is *no* accrued benefit if one attribute converges faster than another.

In the above description of the design sub-space identification algorithm, the issue which remains unresolved is the actual mechanism of identifying any region in design space. In design space, region boundaries are not specified explicitly. Instead, we specify means and standard deviations for all design variables. So, we identify a region in design space in a probabilistic sense. When attribute limits are changed (tightened), new estimates of the means and standard deviations are obtained, as a means of identifying a new corresponding region in design space. The underlying statistical concepts used are described in the following section.

3.3 Basic Statistical Concepts Used

3.3.1 Gaussian Distribution of Variables

In the design assistant of Moses [72], random numbers followed a uniform distribution between the minimum and maximum value limits of a design parameter. This means that there was equal probability of generating all acceptable random values of a variable. This might not always be desirable. For example, for a large motor, the slot dimensions and machine diameters will typically be large. In fact, it makes logical sense that acceptable values for a certain variable should have a certain mean or average value with some variation around this mean. Gaussian Distributions serve our purpose very well. A Gaussian

Distribution has the probability density function given by [50]:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (\sigma > 0) \tag{3.1}$$

where σ is the mean and μ is the standard deviation.

Before we proceed with this assumption, we have to verify if the Gaussian distribution assumption is indeed valid for our data sets. For this purpose, we selected a set of 32 polyphase motors from our industrial sponsors' database. These motor designs were homogeneous in many respects: they all had a cast rotor, pyramidal winding, and a single rotor cage. These motors were all relatively small motors (in the 1Hp - 10Hp range). Each of the design variables were then tested for the normality assumption. Normality assumptions are tested using Chi-squared tests or normal probability plots [50]. We used normal probability plots. Some sample plots are shown in Figure 3-4. It may be seen that plots for virtually all variables fall in a straight line signifying that the normality assumption is valid. In Figure 3-4, the variables shown are: rotor outside diameter, stator slot top width, stator outer diameter, stator slot depth, stator inner diameter, stator slot bottom radius, number of conductors in a turn of wire, rotor slot depth, core length, and number of turns. We observe that for some variables, the plot is not exactly a straight line. However, on the whole, the normality assumption is well justified. We also observe that in the design sub-space algorithm, typically, the variables will be better conditioned (e.g. single horsepower, and speed settings, and tighter limits for machine size), and hence the normality assumption would be more justifiable.

Once we have justified the Gaussian Distribution assumption for all the variables, its implementation in the NDA context is straightforward [96]. Tables for the cumulative distribution may be found in standard Statistics texts like [59] and [50]. These tables are usually for $\mu = 0$ and $\sigma = 1$. A number between 0 and 1, is generated randomly. This number can be used to read the value of the corresponding *normally distributed variable* (Z) from the Normal Distribution Tables. A normally distributed variable X with mean μ and standard deviation σ is obtained as: $X = Z \times \sigma + \mu$.

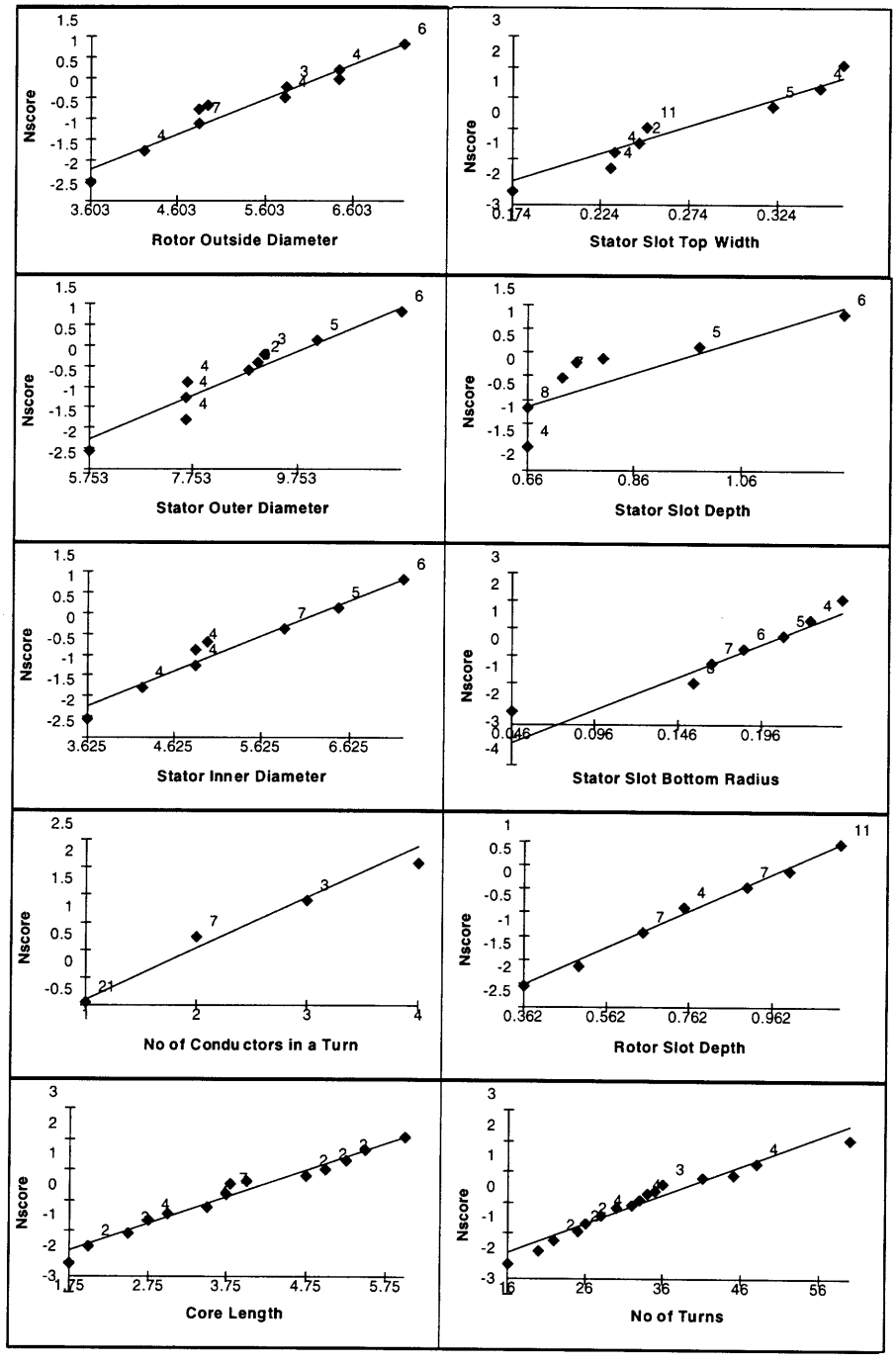


Figure 3-4: Sample Normal Plots for Some Variables

3.3.2 How Multi-Normal Distributions Help

We have verified that all variables in our design space follow a Gaussian (or Normal) distribution. Gaussian distributions have certain desirable properties which makes them very useful for the Design Sub-Space algorithm.

Our aim was to characterize or define a region in $n-$dimensional design space i.e. we are looking for a multivariate description of a region in design space. Multi-Normal distributions (multivariate generalizations of normal distributions) can be used to characterize a multi-dimensional space in a probabilistic sense, and offer two specific advantages over any other distribution.

First, only two parameters (mean and standard deviation) are needed to characterize a univariate normal distribution. This implies that once we obtain the mean and standard deviation, we can characterize the distribution completely. Means and standard deviations are also extremely easy to obtain, and offer easy intuitive interpretations.

Second, the means and standard deviations of all variables, are the *same* parameters which are needed to characterize the multivariate multinormal distribution [69] and [53]. This is an extremely important property, as it allows us to generalize univariate normal distributions to the multivariate multi-normal case with ease. No additional computational effort is required for the multi-normal distribution.

As a result, the Design Sub-Space algorithm now becomes *extremely* simple and straightforward! In order to define or identify a region A_{d1} or A_{d2} in design space, we only need to estimate the means and standard deviations of all the variables!

3.3.3 Means and Standard Deviations

In the design synthesis step of Figure 3-3, we typically generate 200 designs. This number is chosen more or less arbitrarily at this stage. Generating more designs gives us better estimates of means and standard deviations while generating fewer designs in each iteration would undoubtedly make the process faster. Any number which affords a good trade-off would be suitable. We would recommend numbers between 50 and 300. The upper bound of 300 is suggested since the time required for the algorithm increases without any additional

benefit in terms of accuracy. The lower bound of 50 is suggested by [49] for the normality assumption to hold. Our choice of 200 is justified in Chapter 4 where the optimization algorithm requires a large number of data points.

In every iteration of Figure 3-3, after we generate 200 designs, we have to estimate the means and standard deviations in order to characterize the corresponding design space. Sample means and standard deviations are computed directly from these 200 data points. Population means and deviations are then estimated from these numbers. The methodology for estimating population means and standard deviations is explained in standard statistics texts like [50], and is also outlined in [96] which details the implementation in the NDA framework. In short, the sample mean is a good estimate of the population mean, and the population standard deviation is estimated as:

$$\frac{(n-1)S^2}{\chi_2^2} < \sigma^2 < \frac{(n-1)S^2}{\chi_1^2} \tag{3.2}$$

where σ is the population standard deviation, S is the sample standard deviation, n is the number of observations (data points), and the χ^2 values are defined as:

$$P(\chi^2 > \chi_1^2) = 0.995 \quad \text{and} \quad P(\chi^2 > \chi_2^2) = 0.005 \tag{3.3}$$

Then the probability is 0.99 that the χ^2 variable will satisfy:

$$\chi_1^2 < \chi^2 < \chi_2^2. \tag{3.4}$$

This gives us a 99% confidence interval for σ^2.

3.3.4 The Complete Process

Putting it all together, the design sub-space algorithm is given by Figure 3-3 where the design synthesis step is performed as in Figure 3-1, and identification of the region in design space is accomplished by estimating the population means and standard deviations starting from the 200 designs which result from the design synthesis step.

In the following section, we will make some implementation comments and present an example.

3.4 Implementation and Results

3.4.1 Random Numbers

Since the Monte Carlo process relies heavily on the notion of Random Numbers, it is appropriate for us to make a note about how random numbers are generated for our purposes.

We use pseudo–random numbers generated by the computer system. Most C language libraries have a pair of library routines for initializing, and then generating, "random numbers". For details on these routines, the reader is referred to a C language manual like [56].

A very good synopsis of the issues relating to random numbers generation is found in [79]. (For further details, the reader is referred to the references cited therein). It is useful to point out that, here, we *do* rely on the notion of *random* numbers for the Monte Carlo technique to work.

For the purposes of the NDA, the random number generator is seeded by the system clock. For details on the system clock, please see [55]. The result of the above random number generation process is a pseudo–random number between 0 and a (system dependent) maximum. This number is divided by the maximum to produce a number between 0 and 1. This random fraction is used to pick a value of a design parameter between the allowable minimum and maximum limits for that parameter. Section 3.3.1 describes how this random number is mapped onto a normally distributed random number.

3.4.2 Rule Sets

One of the most critical components of the Monte Carlo design synthesis procedure is the analysis routine. Once a design is randomly generated, it is presented to the analysis routine which is used to evaluate the performance. There are usually three outcomes at this stage: the design is rejected because it does not represent a viable design (e.g. when flux densities

are extremely high), the design is theoretically acceptable but the performance is not good, or the design is good and its performance is within (the interim) target limits.

When the design is acceptable, the process runs smoothly. However, especially when the number of design variables is large, very often, a randomly generated design is unacceptable. This happens even when all variables are within reasonable limits, and is the result of interactions between variables. For example, if a motor ends up with extremely small slots, or very low number of turns, air gap flux densities may be large, or slot fill factors may be very high. As a result, a large number of infeasible designs are presented to the analysis routine. This proves to be very inefficient since motor and generator analysis routines may be relatively large and take a significant amount of time to execute (typically each analysis operation takes to the order of a few seconds). In the design sub-space algorithm, we can easily present 10000-20000 designs for evaluation. Hence it is important to reduce the number of infeasible designs generated by the Monte Carlo routine.

There are two ways of improving the process. With the first technique, we can understand some correlations between variables and generate variables with the help of correlations. This approach was followed in [96] where curvilinear regression was used to understand relationships between variables. [84] presents another such technique which could be used for detecting higher-order correlations. [30] outlines a technique for generating correlated random numbers. This technique can be very useful for NDA like systems in general, and is especially valuable when the theory behind the analysis routine is not easily interpretable e.g. analysis routines using simulations or iterative models.

The other technique is to use expert knowledge to ensure that certain interactions between variables are maintained. e.g. No negative geometry etc. The author proposed the first technique in [96] but presently, given that there is ample expert knowledge about the analysis routines, and the number of variables is much larger, we use some expert knowledge in the form of rule sets. These rule sets take the form of simple guiding principles like "No negative geometry", "slot fill $< 100\%$" etc. Rule sets for each solver are detailed in Chapter 4.

By using rule sets, we are sacrificing some generality of the NDA approach. However,

this is justified by the fact that we have developed a more powerful (though application specific) tool for use in an industrial setting.

3.4.3 Design Variable Space and Categorical Variables

It is important to make a couple of quick comments about how categorical variables are handled with the design sub-space identification algorithm.

The design sub-space identification algorithm of Figure 3-3 is sufficiently general to include ordinal (continuous), as well as categorical variables (discrete variables which represent categories and which cannot be ordered). However, the Gaussian distribution implementation works only for ordinal variables. A very simple fix is employed in the NDA. For the NDA, and indeed with motors and generators, there are very few pure categorical variables. Most categorical variables deal with (database) part numbers for certain sub-parts e.g. stator lamination, stator slots, rotor lamination, rotor slots, end rings etc. The user has the flexibility of specifying the part numbers or alternatively, specifying the (ordinal) variables which go on to constitute the sub-part. This simplifies the problem considerably. When the user specifies the constituent variables, the problem is simplified. When the user wants to specify a particular part number, that part number is fixed and the constituent ordinal variables are assumed to be fixed (see below).

The difficulty arises when the user wishes to specify a list of possible part numbers to choose from. In all such cases, when there is a list of possible values that categorical variables can assume, a uniformly distributed random number is used to decide a value for that variable. In subsequent iterations of the sub-space algorithm, this uniformly distributed random number is weighted by the probability of occurrence of a value in the previous run. For example, if the user specified two possible part numbers A and B for a categorical variable, and in the first iteration, 200 designs were chosen, 120 of which had part A and 80 of them had part B, then in the second iteration the probability of choosing part A for a design is $\frac{120}{200}$ or 0.6.

Before we present an example, let us briefly explain what is meant by "fixing" a variable, as mentioned above. This concept will be explained again in Chapter 5.

In the NDA, the user is expected to provide target ranges for all attributes. For each design variable, the user has three choices: 1) Not specify anything, in which case default minimum and maximum limits, and stepsize increments are assumed for that variable. 2) Constrain a variable, in which case, the user specifies new minimum, maximum, and stepsize increment values. 3) Fix a variable, in which case, the user specified fixed value is used for the rest of the process. If the user fixes a variable, effectively, the variable is "removed" (and treated as a constant) for all subsequent steps like the design sub-space identification and optimization.

3.4.4 Example

As an example, we choose to use the MIT 3 Phase Solver, which also happens to be the simplest of all solvers, and is suitable for the first example in this thesis.

Using the MIT solver, our target attributes are: Efficiency $(88 - 100\%)$, Power Factor $(73 - 100\%)$, Iron Mass $(0 - 22Kg)$, and Copper Mass $(0 - 4Kg)$. This is a 3 Hp, 4 pole, 266 V, pyramidally wound motor, with 36 stator slots, 44 rotor slots, fixed rotor radius of $0.04571m$ and a fixed air gap of $0.0003429m$. All other parameters are allowed to range between certain acceptable limits. The aim here was to design rotor and stator slots, number of turns and the core length.

This run was successfully completed (for 100 observations) in about one hour (on an HP 9000 817S). The attribute space is four-dimensional and is difficult to represent on paper. A two-dimensional view of this 4D space is shown in Figure 3-5.

3.4.5 Hit Ratio Improvement

The sub-space identification algorithm, as shown in Figure 3-3 goes through a number of iterations before the attributes converge to the user specified ranges. A natural question arises: Is our sub-space identification algorithm improving its estimate of the design space in every iteration? If the algorithm is indeed successful in identifying a region in the design space, then the design synthesis cycle for the next iteration should proceed relatively faster since we have (incrementally) improved on our estimate of the active region in design space.

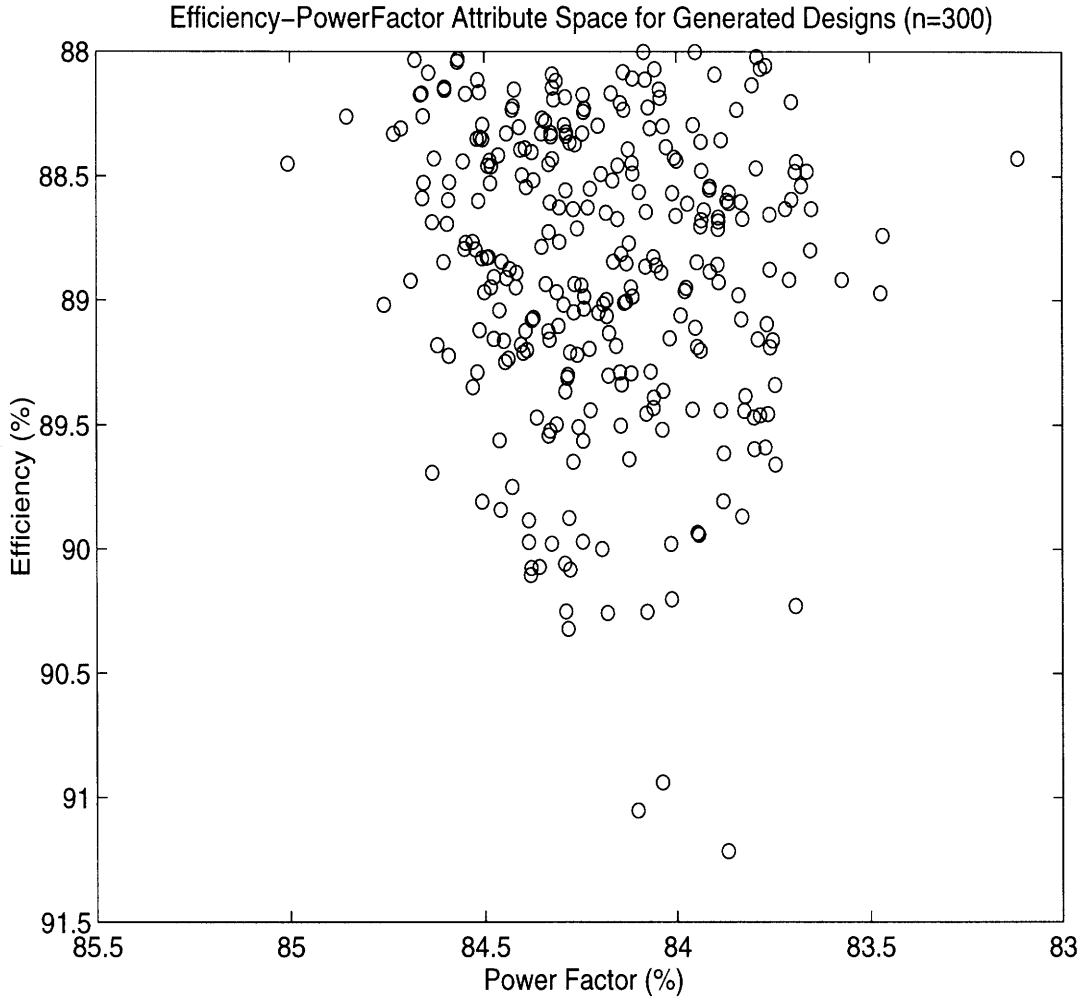


Figure 3-5: A 2D View (Efficiency-PowerFactor) of the Attribute Space

We define a term "hit-ratio" as the ratio of acceptable designs to the total number of designs generated in one design synthesis run. The hit ratio of obtaining an acceptable design during the Monte Carlo design synthesis cycle of Figure 3-1 is a reasonable measure of success and performance of the design sub-space identification algorithm. The hit ratio improvement for every iteration (for the above example) is shown in Figure 3-6. This figure tells us that the hit ratio is small during the first step. This is understandable since at the very outset, the program has no idea about the design space for the current target limits. Thereafter, the hit ratio improves, and remains reasonably high. Even though after the first iteration, the hit ratio is more or less constant, this is a good sign since the performance requirements in every subsequent iteration get tougher and tougher. The fact that the hit ratio does not decrease shows that our design sub-space identification algorithm is proceeding smoothly.

3.5 Recapitulation

This chapter started with a description of the Monte Carlo process, its advantages and disadvantages, and how the Monte Carlo process can be used for design synthesis. Next, we presented our methodology for design sub-space identification. The methodology used some statistical concepts which were outlined next. Finally, in Section 3.4, we made some implementation comments, presented an example and demonstrated that our algorithm performed well in the given situation.

In this chapter, we have used the formulation adopted in Chapter 2 and presented the first step of the proposed methodology. The second step would be outlined in the following chapter.

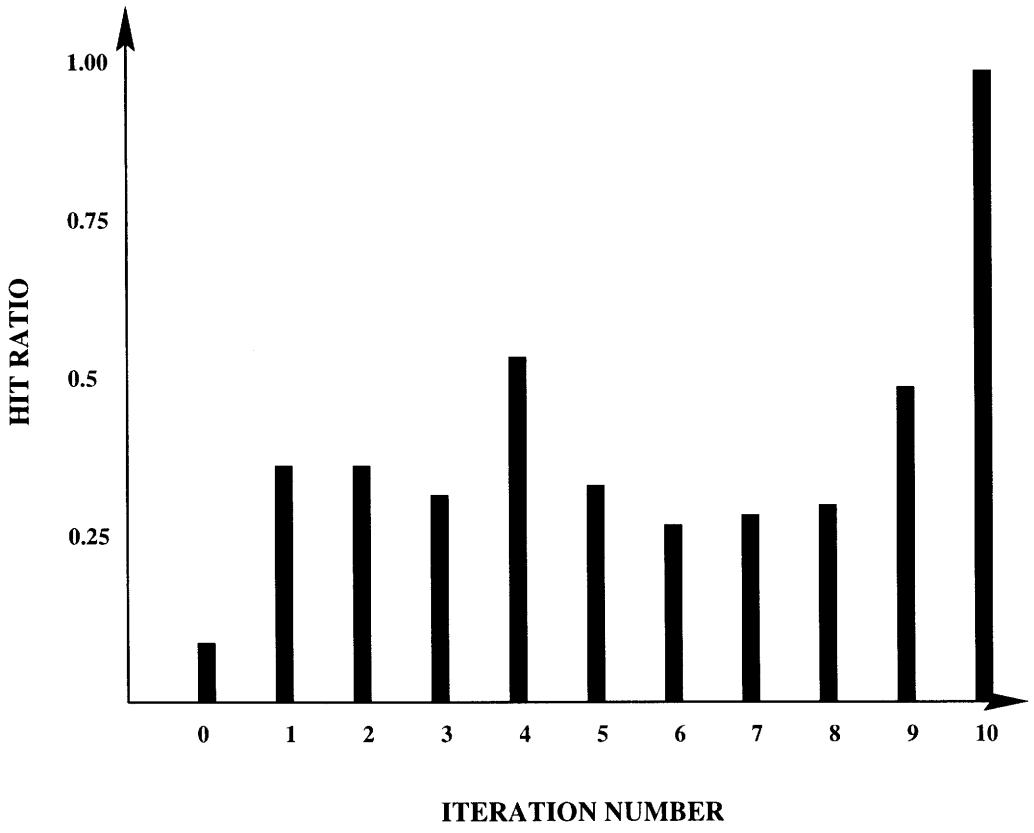


Figure 3-6: Hit Ratio for Every Iteration

Chapter 4

Optimization

The design sub-space identification algorithm lays the ground work for the optimization process. At the end of the design sub-space identification process, we have a few (200) designs which match the user's requirements. This accomplishes one of our goals of facilitating multiple scenario analyses. The user (or designer) can evaluate these designs for any interesting combinations of attributes. At this point, the user may perform simple optimizations by hand or solicit assistance from the NDA. This chapter is devoted to developing the ideas behind the optimization methodology of the NDA. We start by examining the concepts of Multiple Objective Optimization (i.e. optimization when we have multiple attributes). Section 4.2 presents a function evaluation technique which can be very useful for our optimization algorithm. These ideas are tied together in Section 4.3 and results from an example are presented. This example follows from the example presented in Chapter 3. The chapter concludes with a brief summary.

4.1 Multi-Objective Optimization

4.1.1 Introduction

A typical optimization problem has three pieces:

- Design Variables.

- Constraints.

- Objective function.

Design variables are variables for which a suitable set of values has to be decided by the optimization process. This set of variables is the same set of design parameters which constitutes a design e.g. diameter, length of a machine etc. These variables were introduced in Chapter 1 as *Design Parameters*, and the $n-$dimensional space spanned by these variables was shown schematically in Figures 2-2 and 3-2.

Constraints are a set of conditions to be obeyed by design variables. These constraints could take the form of bound constraints (specifying upper and lower bounds on individual variables), equality constraints, or general inequality constraints. In general, both equality and inequality constraints could be linear or non-linear. Constraints represent hyper-surfaces in the design variable space which separate the feasible (acceptable) regions from the infeasible regions.

Objective function is a function of design variables which has to be maximized or minimized. In a multiple-objective optimization scenario, as with the NDA, the objective function is a vector of functions, all of which have to be individually maximized or minimized. The space spanned by all possible values of this vector of functions has been encountered before in Figures 2-2 and 3-2, in the form of a performance variable space. This list of functions is a list of all the *Attributes* (Chapter 1).

Let the vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ be a vector of n design variables. Then the multi-objective optimization problem is formulated as [77]:

Find the vector $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which satisfies the k inequality constraints:

$$g_j(\mathbf{x}) \geq 0 \quad j = 1, 2, \ldots, k \tag{4.1}$$

the p equality constraints:

$$h_j(\mathbf{x}) = 0 \quad j = 1, 2, \ldots, p \tag{4.2}$$

and optimizes the vector function:

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T \tag{4.3}$$

As mentioned above, for the NDA, the f_i represent the different attributes and $\mathbf{x}$ represents the design variables.

After formulating the problem, we are faced with two questions:

- How should we actually perform the optimization?
- How should we choose and decide between alternative solutions?

The first issue, relating to the actual optimization technique to use, is discussed in Sections 4.2 and 4.3 later in this chapter. The second issue, or the decision making problem, is tackled by Multi Criteria Optimization Methodologies and is discussed below.

4.1.2 The Optimal Frontier

Let us consider a hypothetical example with only two attributes: Efficiency and Power Factor. We use two attributes since the performance space for more than two attributes is hard to show on paper. The vector objective function of Equation 4.3 is then:

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x})]^T \tag{4.4}$$

In general, the design vector $\mathbf{x^a}$ which optimizes $f_1(\mathbf{x})$ is different from the design vector $\mathbf{x^b}$ which optimizes $f_2(\mathbf{x})$. There is no way of arriving at a unique answer unless f_1 and f_2 are weighted or prioritized in some way.

Let us say, we assign equal weights to both f_1 and f_2. Here, we are assuming that the numerical values of the two objective functions is comparable, and both the objectives are equally important or valuable. That is to say that we are neutral to the choice between 92% Efficiency & 88% Power Factor, and 88% Efficiency & 92% Power Factor solutions.

For such a case, a unique point may be identified in the performance space and the $\mathbf{x}^*$ corresponding to this point is the optimum design vector. Similarly, if we change the

weights, a different point may be identified in the performance space. In fact, when we look at the performance space, we see a family of designs obtained in this fashion. All of these points in the performance space have the property that no design is superior to the other in terms of *both* attributes. No attribute for any design can be improved without worsening any other attribute. Hence, if no preference or weighting were desirable for the attributes, these points would represent a set of "superior" designs. This family of designs is called the set of "superior", or "non-dominated" or "Pareto optimal" designs (after V. Pareto who formulated this concept of an optimum in 1896). A typical Pareto-optimal frontier in two dimensional attribute space is shown schematically in Figure 4-1. The same idea is easily generalizable to a multi-dimensional attribute space.

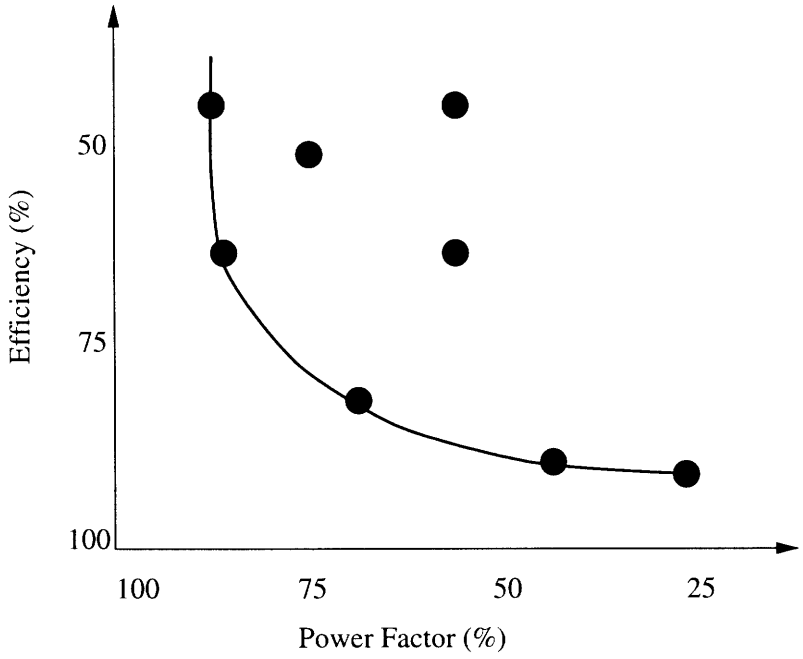


Figure 4-1: The Pareto Optimal Frontier

All points on one side of the frontier (the origin side) are infeasible or unattainable design points. All points on the other side of the frontier are "dominated" designs. For dominated designs, it is possible to find a design which is better with respect to both attributes. In

Figure 4-1, we show three dominated designs, and five non-dominated designs.

Techniques for Multi-Objective Optimization help identify a point on the Pareto frontier based on some logical sequence of steps – usually one which is based on the designer's preferences or domain knowledge.

4.1.3 Techniques for Multi-Objective Optimization

Some simple techniques for multi-objective optimization are presented below. Most of the material below has been adapted from [77] and [29]. It should be mentioned at the very outset however, that in our treatment below, we have not even begun to scratch the surface of this discipline. More detailed descriptions with reference to engineering problems may be found in [77] and [29]. Readers interested in more theoretical and in-depth treatments are referred to [89], [91], [100], and the references therein.

Weighting Objectives Method

We briefly referred to this technique in our example of efficiency vs power factor in the previous subsection. This technique is perhaps the simplest and hence has received a lot of attention for applications. Here, we change our multicriterion optimization problem to a scalar optimization problem by creating one function of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i f_i(\mathbf{x}) \tag{4.5}$$

where $\sum_{i=1}^{k} w_i = 1 \quad w_i \geq 0$ are the weighting coefficients representing the relative importance of the criteria.

Little is known about choosing the weights w_i and the designer has to exercise his intuition to choose an appropriate set of weights. It should be noted that for functions using arbitrary units, these weights do *not* reflect the relative importance of the objectives but are merely factors for locating points on the pareto curve. When the objective functions are represented in units of similar numerical values, these weights represent the relative importance of attributes.

Hierarchical Optimization Method

This considers the case where the attributes can be ordered in terms of importance. We minimize each objective function separately in this order, with each subsequent optimization adding in a new constraint which limits the assumed increase or decrease of the previously considered functions.

Find the minimum $\mathbf{x}^{(1)} = [x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}]^T$ for the primary criterion such that

$$f_1(\mathbf{x}^{(1)}) = min_{\mathbf{x}\in\mathbf{X}} f_1(\mathbf{x}) \tag{4.6}$$

Then for *each* of the remaining objectives, f_i find $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]^T$ such that

$$f_i(\mathbf{x}^{(i)}) = min_{\mathbf{x}\in\mathbf{X}} f_i(\mathbf{x}) \tag{4.7}$$

with additional constraints

$$f_{j-1}(\mathbf{x}) \leq (1 + \frac{\epsilon_j - 1}{100}) f_{j-1}(\mathbf{x}^{(j-1)}) \quad \text{for} \quad j = 2, 3, \ldots, i \tag{4.8}$$

where ϵ_{j-1} are the assumed coefficients of the function increments or decrements given in percent.

Trade-Off Method

In the Trade-Off Method, one objective (the most important one) is treated as the objective function and the others are treated as flexible constraints. The problem then is to find $\mathbf{x}^*$ such that:

$$f_r(\mathbf{x}^*) = min_{\mathbf{x}\in\mathbf{X}} f_r(\mathbf{x}) \tag{4.9}$$

subject to additional constraints of the form

$$f_i(\mathbf{x}) \leq \epsilon_i \quad \text{for} \quad i = 1, 2, \ldots, m \quad \text{and} \quad i \neq r \tag{4.10}$$

where ϵ_i are values of the objective functions which we do not wish to exceed.

Global Criterion Method

Sometimes a vector of attributes is desired which optimizes some global criterion. An example is a function which is a measure of how close we can get to the ideal case (in Figure 4-1 the ideal case is the origin). One common form of this function is:

$$f(\mathbf{x}) = \sum_{i=1}^{k} (\frac{f_i^0 - f_i(\mathbf{x})}{f_i^0})^p \tag{4.11}$$

where $\mathbf{f}^0$ is the ideal solution, and p is usually 1 or 2.

Min-Max Method

The Min-Max formulation differs from the above methods since all the above try to scalarize the objective function in some fashion. The min-max approach (which is a very popular approach) compares relative deviations from the separately attainable minima. In the simplest case, the formulation may be represented as [29]:

$$minimize f(\mathbf{x}) = max_{j=1,\ldots,m}[z_j(\mathbf{x})] \tag{4.12}$$

with

$$z_j(\mathbf{x}) = \frac{f_j(\mathbf{x}) - f_j^0}{f_j^0} \quad j = 1, \ldots, m \tag{4.13}$$

[77] devotes a substantial portion of his book to this technique. Interested readers are referred to [77] for further details.

4.1.4 Chosen Formulation – Trade-Off Method

The choice between the above presented methodologies is largely dependent on the designer. The method which mimics the real situation best should be chosen for an application.

For the NDA, it is extremely difficult to determine the relative importance of variables. Moreover, it is almost impossible to normalize the given attributes to the same scale. (How does one compare Efficiency and Breakdown Torque?) An ordering of the attributes is, of course, out of question. The designer may not always be able to order the attributes

in a clear fashion. Requirements would vary depending on the problem being analyzed. These considerations rule out the Objective Weighting and the Hierarchical Optimization methods. The global criterion and the min-max approaches may be essentially seen as variants of the first two techniques.

The Trade-Off Method on the other hand, seems to follow the NDA situation and formulation very closely. In the NDA, the user begins with *a priori* target limits for all the attributes. For instance, in the example of Chapter 3, the attribute ranges are: Efficiency $(88-100\%)$, Power Factor $(73-100\%)$, Iron Mass $(0-22Kg)$, and Copper Mass $(0-4Kg)$. These limits form the constraints on the attributes. The user can choose *one* attribute for optimization (as the main criterion). In that case, the chosen attribute would be optimized and the others would be maintained within the desired target limits. This mimics real-life situations where we want to maintain these target limits in all scenarios.

The Trade-Off Method for two attributes is shown schematically in Figure 4-2. The desired optimum is shown as a large dot and the attribute limits are shown with dotted lines.

This technique may be conceptually viewed as a case where we "trade away" other attributes (within limits), while improving on the primary criterion.

4.1.5 Considerations

[77] points out that the Trade-Off Method runs into some problems for non-convex problems (like most other optimization techniques), and the final decision is influenced by the starting point chosen. Moreover, there is no analytical technique for determining if a problem (especially a large problem) is convex [77]. Perhaps the only recourse in such a situation is to use multiple starting points.

This highlights two important observations:

- The design sub-space identification and optimization steps make a powerful and useful combination. The design sub-space algorithm gives the user a set of designs to consider, and at the same time, provides a set of starting points for the Optimization step.

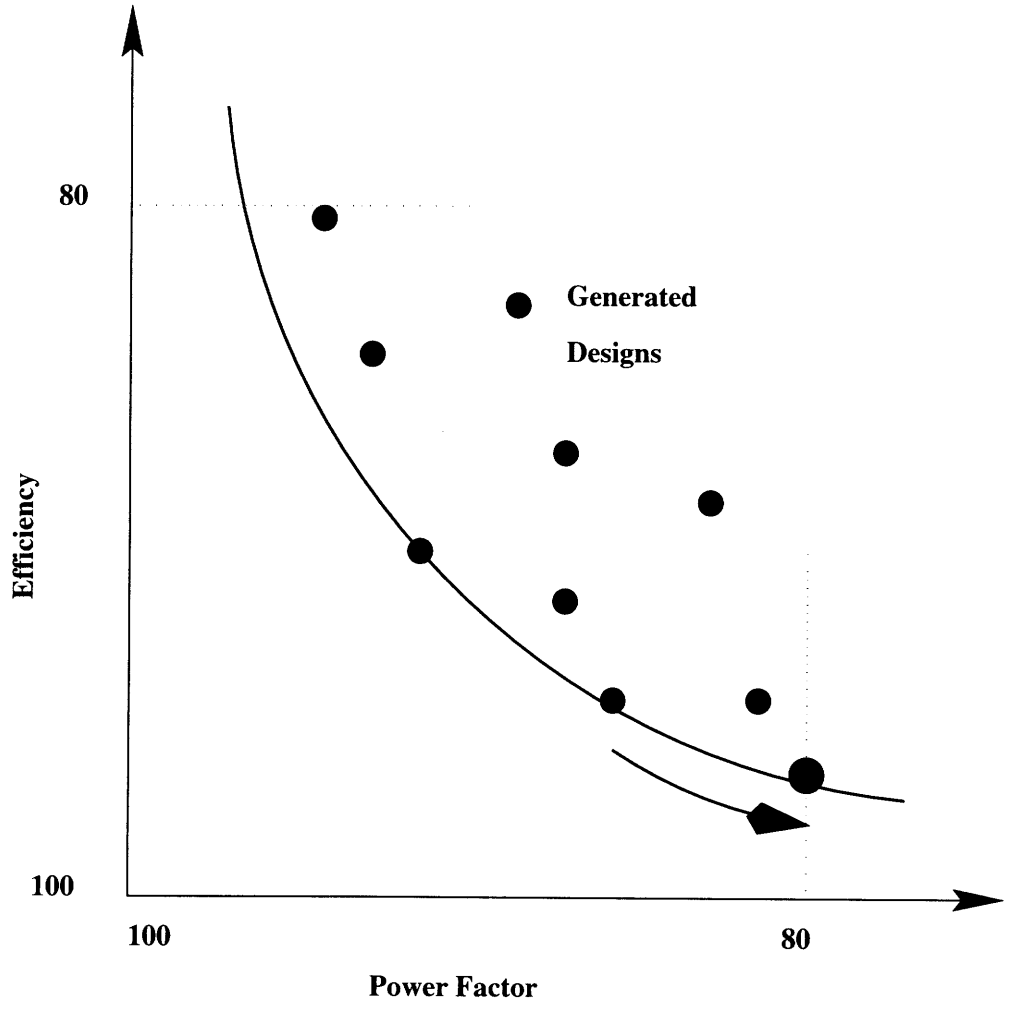


Figure 4-2: The Trade-Off Method.

- Other optimization techniques are not suitable for replacing our two-step methodology for the NDA. No other optimization technique can work without a good set of starting points, and there is no easy way of obtaining these starting points without the sub-space algorithm. And as mentioned in Chapter 2, techniques like Genetic Algorithms which do not care about convexity of the problem are unsuitable for multiple objectives.

4.1.6 Dominance

The concept of Dominance was introduced in Section 4.1.2. Dominance is also discussed in [96] and [72]. In virtually all multi criteria optimization problems, we are interested in non-dominated or superior solutions. However, two important observations have to be made with respect to checks for dominance, before we close this Section on Multi-Objective Optimization.

First, dominance checks are always performed for the complete performance variable space. Lower dimension sub-spaces are not suitable for dominance checks. Let us illustrate this using Figure 4-3, which considers only two attributes: Efficiency and Power Factor.

Points lying on the the solid curve represent superior or pareto optimal designs. The origin represents the ideal scenario, and the points on the other side of the solid curve represent dominated or inferior designs. However, these "inferior" designs need not necessarily be inferior if there are additional attributes of interest. For example, in the NDA, there are other attributes of interest like breakdown torque, iron volume (mass) etc. In such a situation, a point which appears to be inferior in the Efficiency-PowerFactor space (2D view of the multi-attribute space) may lie on the frontier in the complete attribute space.

This can be seen in Figure 3-5 where most of the points appear to be inferior in this 2D view. However, they need not necessarily be so when all four attributes are considered. Hence, checks for dominance must be performed for the *complete* attribute space.

This brings us to the second observation. The *"complete"* attribute space is again an idealization as far as dominance checks are concerned! We are assuming that our set of attributes constitutes the *only* attributes that a designer would be interested in considering

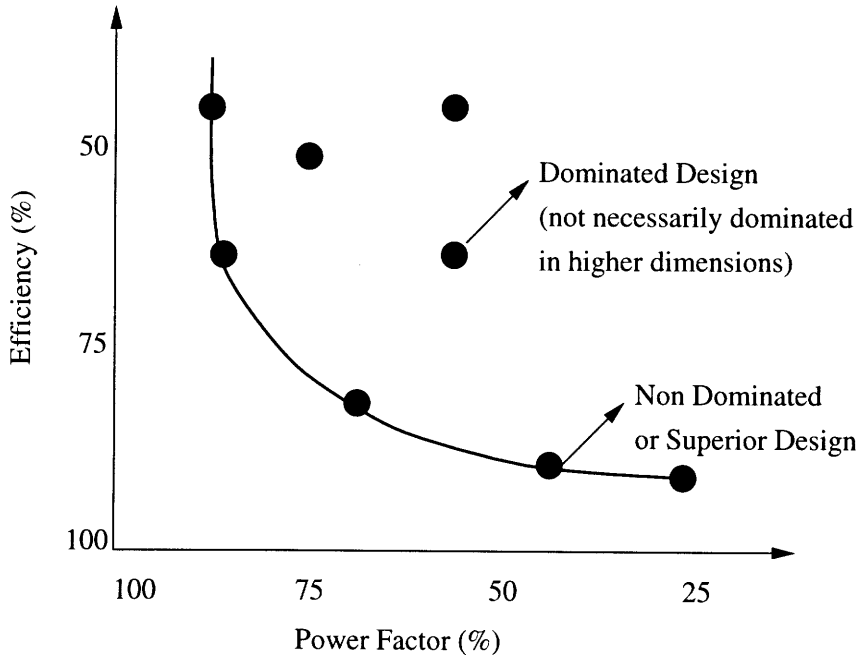


Figure 4-3: The Dominant Frontier in 2D Attribute Space

while evaluating a design. Let us examine two scenarios here.

In the first case, let us say we drop one of the attributes in the example of Chapter 3. Understandably, may designs would now be inferior when evaluated with respect to the remaining three attributes. Does that mean that we can throw away those designs? The answer of course, depends on the situation. Sometimes, those designs may still be of interest to the customer. From a practical stand-point, just because we dropped one attribute from our design procedure, it does not mean that we have lost all interest in that attribute altogether. This also leads to our second scenario.

Even when we have a "complete" set of attributes, sometimes additional factors may be of interest to the designer or customer. These factors could be design variables, or additional attributes not considered in our initial set of attributes. For example, in a polyphase motor design, the attributes used in the NDA (discussed in Chapter 6) are Efficiency, Power Factor, Breakdown Torque, Locked Rotor Torque, Locked Rotor Current, Slot Fill and Core Volume. Even though this list is fairly complete for most applications of polyphase motors, frequently, other attributes like copper weight, and current densities in different parts of the machine, are important. The list of such additional attributes is in fact pretty long. In a real-life industrial setting, an interesting case was observed. A factor which was not even an "attribute" gained prominence based on the application of the motor. A 3 Hp 4 pole polyphase motor was being designed (using the NDA) for a compressor application. For compressor applications, it is important to maintain a constant speed. The ability of a design to meet a specified target speed became an important consideration. Target speed is not even an attribute according to our definition (where we defined attributes as parameters which were always minimized or maximized)! In another interesting case, the NDA produced a set of designs (matching a set of user requirements), one of which had an unusually large air gap. Its attributes, however, were satisfactory. This design is of immense interest to a motor designer. Large air gaps mean lower manufacturing costs! Of course, there are some problems associated with unusually large air gaps (like increased magnetizing current and its related losses, increased "humming" sound if the motor is rigidly mounted, etc). But there are some distinct advantages, like lower cost, lower stray-load losses, and a possible increase

in breakdown torque [115]. Lower cost in itself makes the design interesting enough from an industrial stand-point. Designers would be interested in exploring this design further. In this case, a *Design Parameter* was important in distinguishing a design.

These observations underscore two important points:

- For an industrial strength solver and design situations, designs are perhaps better *not* rejected on dominance considerations alone. Of course, there are some disadvantages to this: invariably truly inferior designs creep into a set of generated designs, and many designs turn out to be similar. But when the list of "possible" attributes is large, it is perhaps not a high price to pay. This recommendation, however, is only for NDA-like design tools. In general, dominance check remains important concept and designers have to keep it in mind in any multi attribute design situation.

- The set of attributes can never be considered to be "complete". Hence for performing a multiple scenario analysis, the resulting designs from our design sub-space identification algorithm are extremely important! Of course, the user would benefit from additional software tools which help navigate through the large number of presented designs efficiently.

4.2 MARS as a Function Approximation Technique

In Section 4.1.4, we discussed our multi-objective optimization strategy of choice. However, we still need a technique to actually perform our constrained optimization. Many optimization techniques like Genetic Algorithms, Simulated Annealing, conventional optimization and neural networks have been outlined in Chapter 2. Hence, we will not repeat the discussion here. In this section, we will change gears to discuss a function approximation technique which we use to help us with optimization. The ideas from Section 4.1 and this Section are tied together in Section 4.3.

4.2.1 How These Techniques Help with Optimization

Function approximation techniques do not directly perform optimization. However, they can be valuable tools in certain scenarios when they are used as part of optimization processes.

Most regular optimization processes require two capabilities: the ability to compute the value of a function at a specified point, and the ability to obtain the derivative of the function at a desired point. There are certain techniques like Genetic Algorithms which do not require derivatives but they require an extremely large number of function evaluations. As discussed before, we are not using Genetic Algorithms in this thesis. Other conventional optimization techniques also exist which do not require derivatives but require a large number of function evaluations. These do not tend to be as powerful as conventional optimization techniques which require derivatives.

Most analysis routines (Figure 2-2) consist of complicated circuit models and can be extremely large. As explained in Chapter 2, these usually comprise a complex set of equations which cannot be written explicitly. Obtaining derivatives for these equations is out of question. No analytical techniques exist for obtaining derivatives of such equations. Moreover, these circuit equations can be extremely large and may require a long time for one computation. An obvious example is models based on Finite Element Analysis, which take an extremely long time for one computation. They also do not provide us with any analytical form of the relationships. Hence obtaining derivatives is still an open question.

Function approximation techniques help us in such situations. These techniques take in sets of input and output data, and approximate a relationship between them. Such models are extremely fast to compute (once the function has been approximated), and in certain cases, also provide us with analytical forms of the relationship which could be used to estimate derivatives.

Say the set of inputs or explanatory variables (or design variables in the case of the NDA) is denoted by a vector $\mathbf{x}$, and one output is denoted by y. Then if the functional relationship between y and $\mathbf{x}$ is:

$$y = f(\mathbf{x}) \tag{4.14}$$

then the function approximators attempt to model the function f:

$$y = \hat{f}(\mathbf{x}) \tag{4.15}$$

Neural networks have been discussed before as function approximation techniques. However, they do not provide us with analytical forms of the approximated relationship. This makes them unsuitable for our purposes. There are some other subtle issues associated with Neural networks. They model a global fit, and hence, local behavior sometimes cannot be modeled. Some such issues have been examined in [92]. Neural networks, however, have one clear advantage over many other techniques: they can be used to model Multiple Input Multiple Output systems. Hence in Equation 4.15, y could be a vector of outputs $\mathbf{y}$.

Our technique of choice is MARS (Multivariate Adaptive Regression Splines). It is a statistical technique which performs non-parametric regression to give us a model of the relationship. This analytical relationship can also be used to obtain derivatives. We will devote the rest of this section to MARS. MARS and Neural networks make an interesting comparison. [24] is devoted to comparing these two estimation techniques and shows that MARS comes out to be superior to Neural Networks in many respects including speed and performance.

At this stage, we must also point out that there are other function approximation techniques found in statistics literature. Their use and applicability depends on the problem at hand. The reader is referred to [14], [42], [43], [39], [48] and to the discussions in [41] for a few of these methods.

4.2.2 Classification and Regression Trees

To obtain an understanding of MARS, let us first start with the precursor of MARS, called Classification and Regression Trees or CART [1]. Both CART and MARS currently handle only multiple input single output (MISO) systems. A good synopsis of CART and MARS (with a MARS based application) may be found in [92]. Classification and Regression Trees

[1]CART is a trademark of California Statistical Software Inc.

are described fully in a text by the same name [15].

CART, as the name suggests, classifies the data and performs a regression on the data. It recursively partitions the data in a binary tree like fashion and performs a (different) regression on each of the partitions. Figure 4-4 shows the recursive partitioning schematically. Here, we have assumed that there are only two design variables x_1 and x_2, and one performance variable y. We have N observations or data points, with these three pieces of data each.

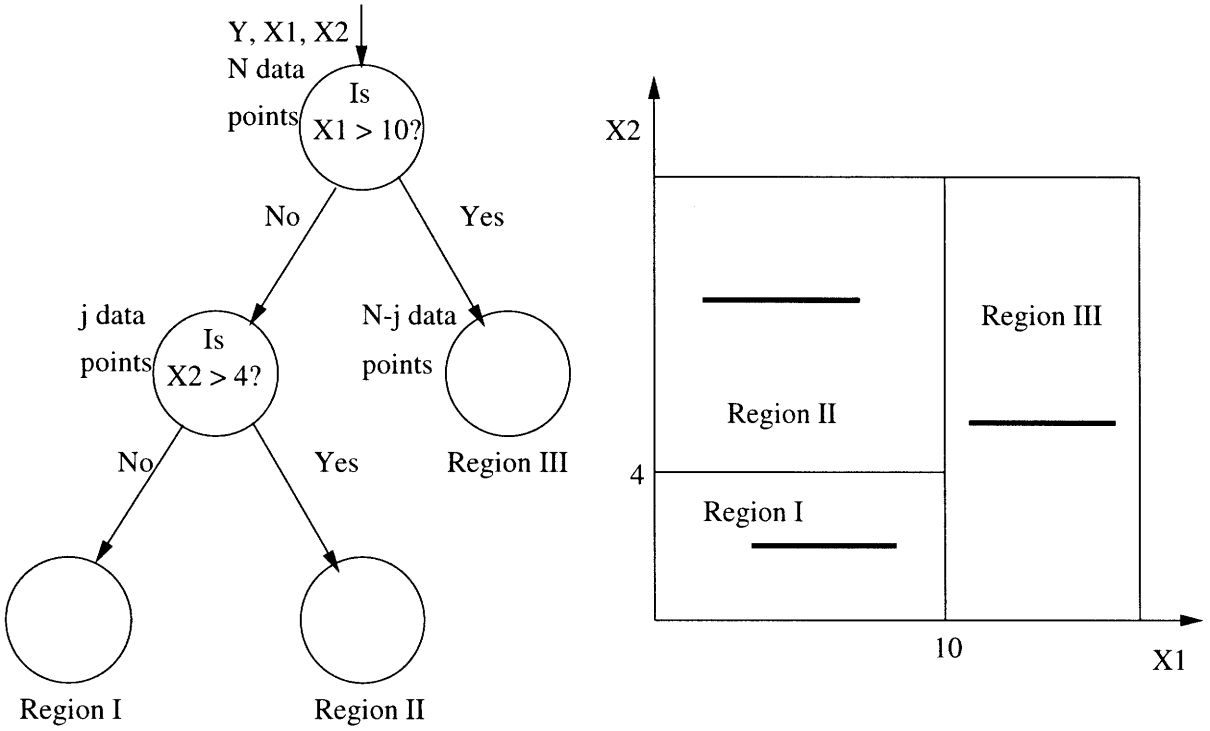


Figure 4-4: Classification and Regression Trees

At the first node, we have a query which determines if $x_1 > 10$. At this stage, there are two outcomes, *yes* and *no*. j out of N data points, which answer *no* go into the left branch of the binary tree and the rest $N - j$ go into the right branch. The left branch is further partitioned based on another query $x_2 > 4$. This results in three distinct regions as shown

in Figure 4-4. The queries and the "cut-off" points (10 for x_1 and 4 for x_2) are ascertained from the data (which includes the response y). In general, recursive partitioning continues until a pre-defined criteria is met. The resulting tree is then pruned to develop parsimony in the model. Regression is performed in the resulting regions.

This regression typically takes the form of an expansion in a set of basis functions:

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} a_m B_m(\mathbf{x}) \tag{4.16}$$

where M is the total number of partitions created, and a_m are the coefficients of expansion. (CART uses data to determine the regions, and to estimate the coefficient values.) The basis functions B_m take the form:

$$B_m(\mathbf{x}) = \prod_{k=1}^{K_m} H[s_{km} \cdot (x_{v(k,m)} - t_{km})] \tag{4.17}$$

where K_m is the number of splits that gave rise to B_m, s_{km} take on values of ± 1 indicating the right/left child at the kth non-terminal node, $v(k,m)$ label the predictor variables at the kth non-terminal node, t_{km} represent values (knot locations) on those variables, and H is the step function indicating a positive argument

$$H[\eta] = \begin{cases} 1 \ \ if \ \eta \geq 0, \\ 0 \ \ otherwise \end{cases} \tag{4.18}$$

CART creates rectangular non-overlapping regions, and $\hat{f}(\mathbf{x})$ has discontinuities due to jumps at boundaries of these non-overlapping regions. Due to recursive partitioning, CART cannot create additive models. Deleting a node leaves a hole in the predictor space. Most of these shortcomings are overcome using MARS models.

4.2.3 MultiVariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines (MARS) is a modification of CART. It overcomes some limitations of CART like inability to create additive models and regression function

discontinuities at region boundaries. MARS partitions any current candidate node or any of its predecessors. This creates overlapping regions in space and facilitates additive modeling. In addition, splines are used to replace step functions, and this addresses continuity issues.

MARS was developed by Professor Jerome H. Friedman of Stanford University, and is fully described in [41] and the discussions therein. Extensions of MARS to handle categorical data sets is described in [40]. [90] offers a tutorial to help learn about MARS, and [92] and [64] present examples of applications of MARS to a semiconductor manufacturing application, and to modeling time series respectively.

CART fitted constant functions within each subregion. MARS by contrast, uses splines. This leads to better regression models and offers better continuity properties at region boundaries. (For an introduction to splines, see [23]. [31] offers a more theoretical treatment of spline smoothing and non-parametric regression.) MARS basis functions consist of one sided or two sided truncated power basis functions for representing qth order splines.

$$b_q(x-t) = (x-t)_+^q \tag{4.19}$$

is a one-sided truncated power power basis function. A two-sided truncated power basis is a mixture of functions of the form

$$b_q^{\pm}(x-t) = [\pm(x-t)]_+^q \tag{4.20}$$

where t is the knot location, q is the order of the spline, and the subscript indicates the positive part of the argument. For $q > 0$, the spline approximation is continuous and has $q-1$ continuous derivatives. The step functions appearing in CART may now be seen to be two-sided truncated power basis functions for $q = 0$ splines.

Similar to Equation 4.17, the basis functions then take the form

$$B_m^{(q)}(\mathbf{x}) = \prod_{k=1}^{K_m} [s_{km} \cdot (x_{v(k,m)} - t_{km})]_+^q \tag{4.21}$$

Another limitation of CART was its inability to model additive and linear effects. This

is overcome in MARS by allowing all nodes (including all non-terminal nodes) and the root node to be eligible for further splitting. Splitting the root node allows for modeling additive effects easily. High and low order interactions are also easier to model in this framework.

As a result of these changes, MARS cannot be visualized as a tree. It allows overlapping regions with regression models having appropriate continuity properties at region boundaries. As a result, even a schematic is difficult to show in two-dimensions. A schematic is shown in Figure 4-5 which highlights the fact that MARS has *different* regression models in different regions, and the regions are *overlapping*. Continuity is not depicted in this schematic representation.

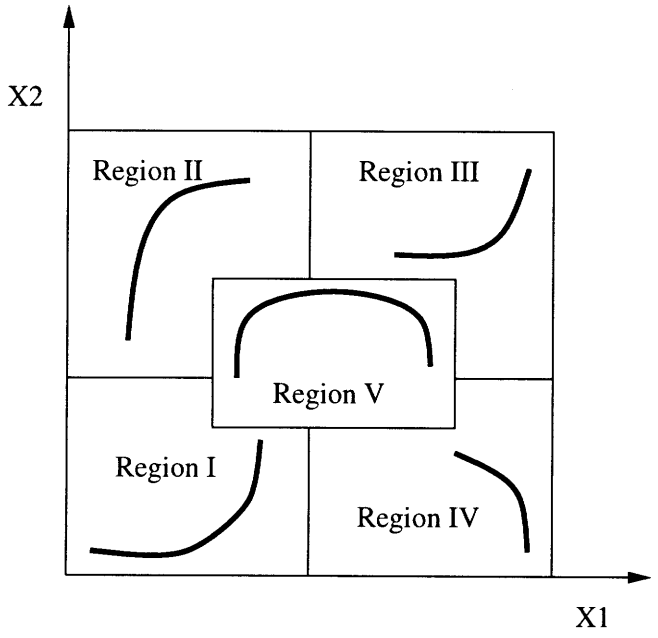


Figure 4-5: MARS Schematic

In the NDA context, MARS is used to approximate functions in multiple dimensions. Understandably, these are hard to represent on paper. However, lower dimensional "slices" can be printed on paper. This functionality is provided with the MARS software. One such sample plot of a MARS surface is shown in Figure 4-6. This figure shows a 3D plot of

efficiency vs. rated power and slot fraction.

4.2.4 Gradients from MARS Models

The final mars model then takes the form [41]

$$y = \hat{f}(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m \prod_{k=1}^{K_m} [s_{km} \cdot (x_{v(k,m)} - t_{km})]_+ \tag{4.22}$$

using splines with order $q = 1$.

This model can also be recast into the form

$$y = \hat{f}(\mathbf{x}) = a_0 + \sum_{K_m=1} f_i(x_i) + \sum_{K_m=2} f_{ij}(x_i, x_j) + \sum_{K_m=3} f_{ijk}(x_i, x_j, x_k) + \cdots \tag{4.23}$$

where the first sum is over all basis functions that involve only a single variable, the second is over all basis functions that involve exactly two variables, and so on.

For the purposes of the NDA (and for most MARS implementations [41]), we limit ourselves to a maximum of only two variable interactions.

One of the important issues in a MARS model is the degree of continuity to impose on the fit i.e. the order of the spline q. There are statistical and computational tradeoffs ([41]) which lead us to use $q = 1$ splines in Equation 4.22, and to use the resulting solution with discontinuous derivatives to derive a continuous derivative solution.

The strategy employed in [41] is to replace each spline function (Equation 4.20) with a corresponding truncated cubic function of the form:

$$C(x|s = +1, t_-, t, t_+) = \begin{cases} 0 & x \le t_-, \\ p_+(x - t_-)^2 + r_+(x - t_-)^3 & t_- < x < t_+, \\ x - t & x \ge t_+, \end{cases} \tag{4.24}$$

$$C(x|s = -1, t_-, t, t_+) = \begin{cases} -(x - t) & x \le t_-, \\ p_-(x - t_+)^2 + r_-(x - t_+)^3 & t_- < x < t_+, \\ 0 & x \ge t_+, \end{cases} \tag{4.25}$$

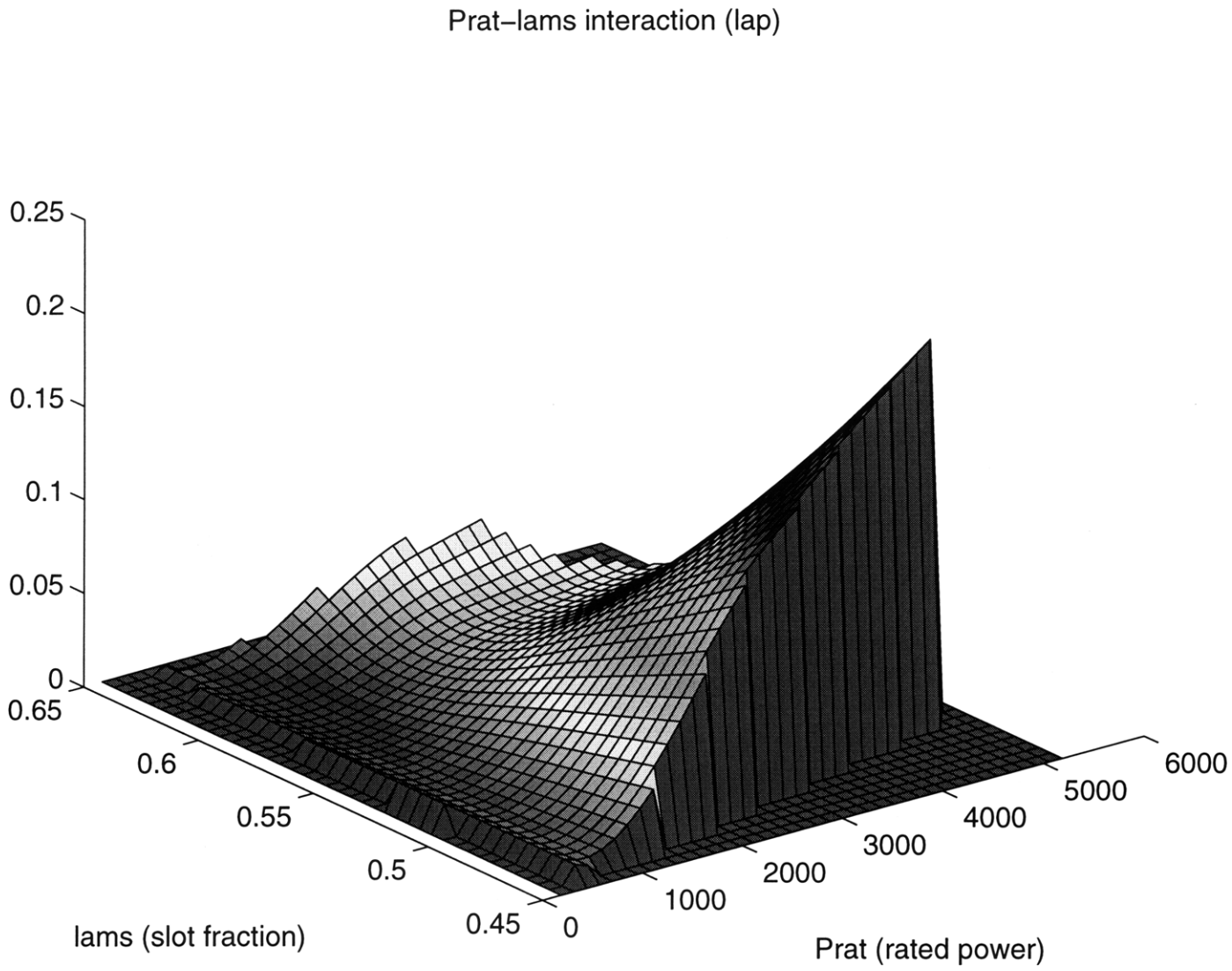


Figure 4-6: A Sample MARS Surface Fit

with $t_- < t < t_+$. Setting

$$\begin{aligned} p_+ &= (2t_+ + t_- - 3t)/(t_+ - t_-)^2, \\ r_+ &= (2t - t_+ - t_-)/(t_+ - t_-)^3, \\ p_- &= (3t - 2t_- - t_+)/(t_- - t_+)^2, \\ r_- &= (t_- + t_+ - 2t)/(t_- - t_+)^3, \end{aligned} \tag{4.26}$$

causes $C(x|s, t_-, t, t_+)$ to be continuous and have continuous first derivatives. There are second derivative discontinuities at $x = t_\pm$.

Derivatives are then obtained from the MARS model of Equation 4.23:

$$y' = \hat{f}'(\mathbf{x}) = \sum_{K_m=1} f_i'(x_i) + \sum_{K_m=2} f_{ij}'(x_i, x_j) \tag{4.27}$$

where the derivatives for the truncated cubic functions in Equation 4.25 are obtained as:

$$C(x|s=+1, t_-, t, t_+) = \begin{cases} 0 & x \le t_-, \\ 2p_+(x - t_-) + 3r_+(x - t_-)^2 & t_- < x < t_+, \\ 1 & x \ge t_+, \end{cases} \tag{4.28}$$

$$C(x|s=-1, t_-, t, t_+) = \begin{cases} -1 & x \le t_-, \\ 2p_-(x - t_+) + 3r_-(x - t_+)^2 & t_- < x < t_+, \\ 0 & x \ge t_+, \end{cases} \tag{4.29}$$

This then helps us obtain the gradient of an approximated function using MARS.

4.3 Proposed Methodology

In Section 4.1, we discussed multi-criteria optimization, and in Section 4.2 we presented a technique for obtaining gradients from an approximated function. The only missing piece is an algorithm for actually performing the optimization. This is accomplished using one of the popular non-linear programming techniques. The software used is called *donlp2*.

4.3.1 Implementation

The optimization algorithm is implemented using the following sequence of steps:

1. We formulate our optimization problem as shown in Equations 4.9 and 4.10.

2. The user selects a primary criterion which is used as the objective function. All other attributes are incorporated as constraints with bounds which were initially supplied by the user before the sub-space identification step.

3. The 200 (or appropriate number of) data points from the design sub-space identification step are used to compute the MARS model.

4. Optimization is then performed using a non-linear programming tool called *donlp2*.

5. *donlp2* requires a feasible starting point. Each of the 200 starting points from design sub-space identification are used as starting points. Hence, the optimization process is run 200 times with different starting points.

6. For each function evaluation, *donlp2* invokes the solver, and for each gradient, it calls an appropriate routine which uses expressions for gradients obtained using MARS models.

7. Finally, the optimum solution is presented to the user.

Both MARS and *donlp2* are softwares available in the public domain. MARS was developed by Professor Jerome H. Friedman (*jhf@stat.stanford.edu*), and the software is obtained from (*http://lib.stat.cmu.edu/general/*). *donlp2* was developed by Professor Peter Spellucci (*spellucci@mathematik.th-darmstadt.de*), and is available by ftp from *ftp://netlib.bell-labs.com/netlib/opt/donlp2.tar*. It can also be obtained from *http://plato.la.asu.edu/donlp2.html*.

4.3.2 Considerations and Limitations

Our optimization approach consists of using the Trade-Off Method to perform multiple objective optimization. The actual optimization is carried out using a non-linear program-

ming tool called *donlp2*, and objective function and constraint gradients are computed using MARS models.

Let us summarize some of our considerations for developing a suitable optimization algorithm, and how we dealt with them using our proposed optimization methodology.

- **Multiple Objectives**. We have investigated a number of multi objective optimization procedures, and decided to use the Trade-Off Method, which seems to be best suited for our case.

- **Large Number of Variables**. These are handled very well by the design sub-space identification algorithm. Moreover, *donlp2* handles a large number of variables and constraints. MARS is not limited by the large number of variables either.

- **Costly Evaluation of $f(x)$**. In the trade-off method, the primary attribute constitutes the objective function, and other attributes form non-linear inequality constraints. Our circuit models (solvers) are used to compute values of all attributes. Even though such circuit model analyses are expensive (each analysis takes to the order of a few seconds), we use them instead of MARS function approximation estimates to enhance the accuracy of our optimization process. As a result, the NDA runs for a few extra hours but accuracy considerations justify the extra running time.

- **No function gradients**. This is solved using MARS. Gradients are estimated using MARS models.

- **Non-linear constraints**. These constraints are handled well in the *donlp2* framework. This is one of the most powerful features of using conventional optimization.

- **Non-linearities in the space**. This does pose a problem for the optimization step. However, non-linear programming tools like *donlp2* handle these with satisfactory results.

- **Discreteness**. This is perhaps the biggest concern with our proposed methodology. Discreteness was handled well at the design sub-space identification step. However, at

the optimization step, *donlp2* and other non-linear programming tools do not handle discreteness gracefully. We tackle this potential problem by using multiple starting points for the optimization process.

- **Categorical Variables.** An issue related to discreteness is handling of categorical variables. For the NDA, since the number of pure categorical variables is low, we "fix" the categorical variables for the optimization run. Categorical variables were handled in the design sub-space identification step. Hence, the 200 sample points include designs with potentially different categorical variable combinations. Since we use each of these designs as starting points, we effectively start with potentially different categorical variable combinations. Categorical variables are then fixed for the rest of the optimization process (their values are dictated by the starting point).

We would also like to mention an implementation detail associated with handling discrete variables. Formally, discrete variables are handled in optimization processes by employing additional equality constraints. For example, if a variable x can only assume two discrete values: 1 and 2, then the corresponding equality constraint is:

$$(x-1)(x-2)=0 \tag{4.30}$$

However, if the number of such discrete variables is large, these constraints can really hamper the accuracy and performance of *donlp2* (or any other optimization algorithm). Hence, in the NDA, we do not employ these equality constraints. Instead, we treat discrete variables as continuous variables but limit the values they can assume to the allowable set of discrete values. This process has the disadvantage that the optimization process could potentially yield local optimum solutions. This is handled by the fact that we are using multiple starting points for our optimization process, and there is a very good likelihood of attaining the global optimum. This idea is displayed graphically in Figure 4-7.

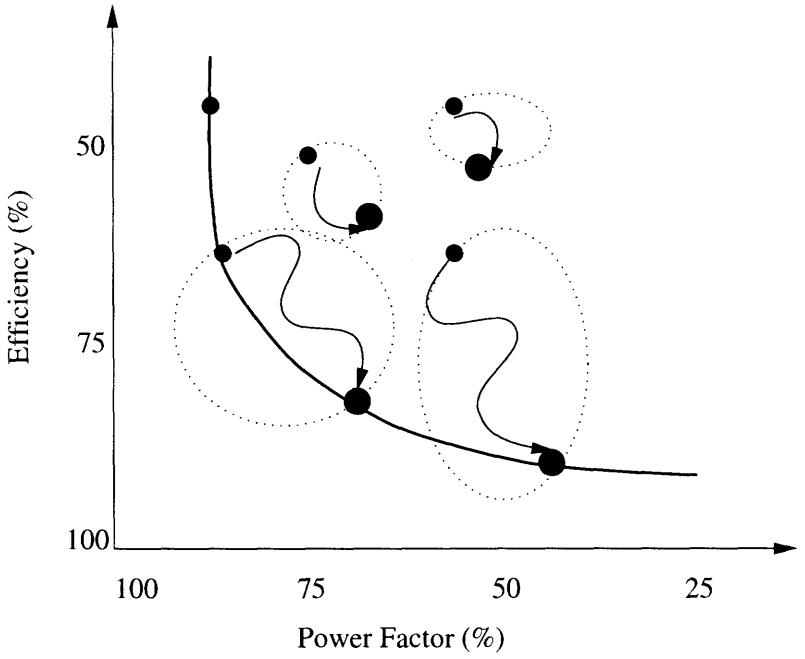


Figure 4-7: Limitations of our Proposed Methodology

4.3.3 Advantages of Combining with Sub-Space Identification

As has been mentioned throughout this chapter, our optimization algorithm is powerful in part because it is used in conjunction with the design sub-space identification algorithm described in Chapter 3. Here, we will summarize some of the advantages of having an integrated optimization methodology using the design sub-space identification algorithm.

- Design Sub-Space Algorithm facilitates a multiple scenario analysis, and lets the user consider a wide variety of design solutions.
- Provides the data points for MARS.
- Handles some of the limitations of the Trade-Off Method by supplying multiple starting points.
- Handles problems associated with discreteness in the optimization step by supplying multiple starting points.
- Provides *feasible* starting points for the optimization algorithm, all of which satisfy all the constraints of the formulated optimization problem.
- Provides an easy solution to the problem of dealing with categorical variables.

4.3.4 Example and Results

Let us revisit the example used in Chapter 3 and apply the optimization algorithm presented in this chapter to that example. The example used the MIT solver and had target attributes of: Efficiency $(88 - 100\%)$, Power Factor $(73 - 100\%)$, Iron Mass $(0 - 22Kg)$, and Copper Mass $(0 - 4Kg)$. The design was for a 3 Hp, 4 pole, 266 V, pyramidally wound motor, with 36 stator slots, 44 rotor slots, fixed rotor radius of $0.04571m$ and a fixed air gap of $0.0003429m$. All other parameters were allowed to range between certain acceptable limits. The aim was to design rotor and stator slots, number of turns and the core length. Our results from the design sub-space identification step were shown in Figure 3-5.

Optimization was performed using this dataset. Running the optimization algorithm to the data obtained from the previous step took an additional three hours (on an HP 9000

817S). The two dimensional Efficiency-PowerFactor attribute space is shown in Figure 4-8. Both generated designs and designs resulting after optimization are shown on this figure. Please note that the scale used in Figure 4-8 is different from Figure 3-5.

The best optimized design obtained had an Efficiency of 92.65%, up 3.63% from the design that it started with before optimization. For this design, the Power Factor was 82.45% (down 2.31%), the Iron Mass was $19.8Kg$ (up $0.62Kg$), and the copper mass was $1.41Kg$ (up $0.564Kg$). The higher efficiency was achieved primarily by making slots smaller and the core length and slot fill larger compared to the starting design.

This result illustrates the Trade-Off Method vividly. In order to achieve higher efficiency, all other attributes are worse off but are still within user specified limits. The reader might recall that for a motor design, higher efficiency and power factor, and lower iron mass and copper mass are desirable. So, in a sense, we have "traded away" some of the other attributes (within limits), and have used that to "buy" more Efficiency!

Points in Figure 4-8 also seem to lie on a discernible pareto frontier.

For the entire set, the average improvement in Efficiency was 2.451%, and the maximum improvement in Efficiency for any one starting point was 4.1%. This is shown in Figure 4-9.

We must also mention at this point that these results are obtained from the MIT solver which is not an industrial strength solver. Such remarkable results should not be expected from real-life industrial strength solvers especially when we start with a production design and allow only a few parameters to vary. We will present some examples with industrial solvers in Chapter 6.

4.4 Recapitulation

In this chapter, we have presented the ideas behind multiple objective optimization and how they apply to the NDA. Two issues which emerge from this subject are: how to perform the optimization, and how to evaluate the quality of the solution. Evaluation is performed using the Trade-Off Method where we optimize for one attribute while treating other attributes as non-linear constraints. The actual optimization is performed using a conventional non-linear programming tool (*donlp2*), and a statistical function approximating tool called MARS

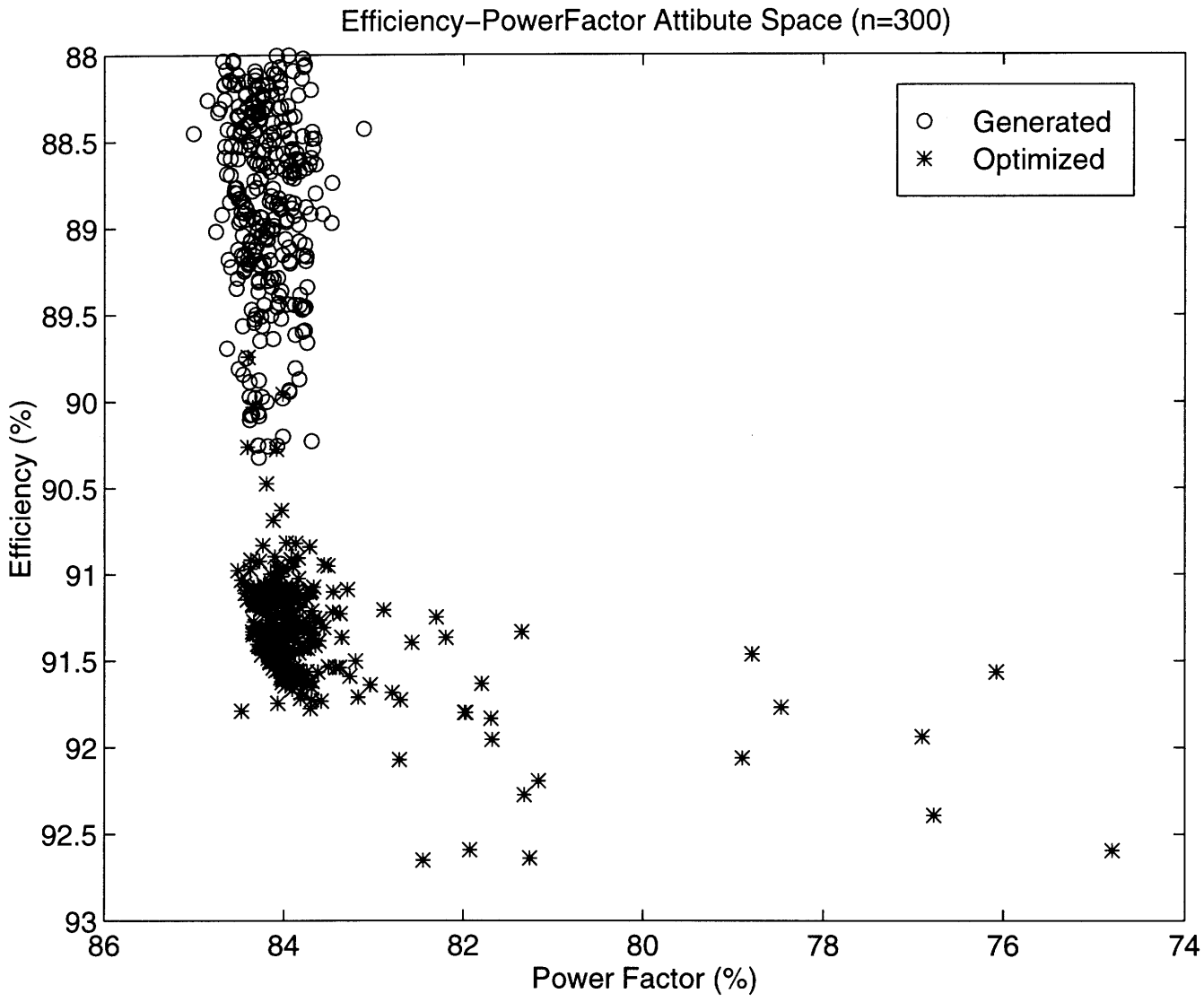


Figure 4-8: A 2D View of the 4D Attribute Space with Optimized Designs

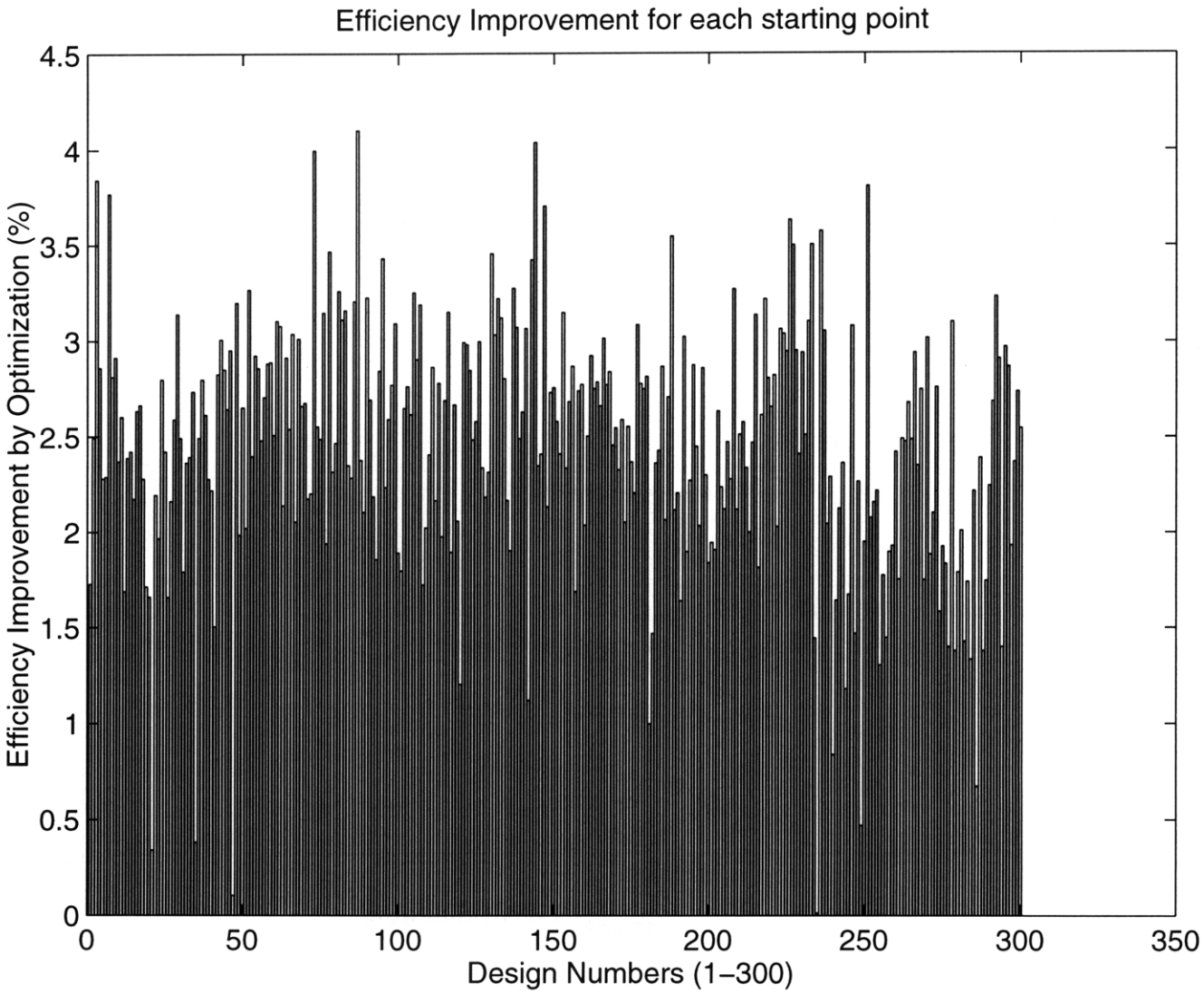


Figure 4-9: Efficiency Improvement for Every Starting Point

which is used to obtain function gradients. In other applications, where analysis through circuit (or other) models is expensive e.g FEM, MARS may be used for an estimate of the function value also. These ideas are tied up in Section 4.3 which also takes the example from Chapter 3 and presents optimization results for it.

At this stage, we have presented our two-step methodology and illustrated it with an example using the MIT solver. What remains is the implementation of this methodology to produce an industrial strength software system. This is presented in Chapter 5. Chapter 6 will present some additional results using other industrial solvers.

Chapter 5

Implementation and Software System

So far in this thesis, we have formulated our problem (Chapter 2), and presented a two-step methodology for solving the formulated problem (Chapters 3 and 4). We also presented an example to illustrate our proposed methodology. In this chapter, we change gears to explain how these ideas are implemented to develop an industrial strength software system. We begin with a description of the software system, its inputs, outputs, and organization, in Section 5.1. Some of the input variables are explained with the help of diagrams in Section 5.2. In Section 5.3 we briefly describe a graphical user interface used to run our software. Earlier, in Chapter 3 and Chapter 4, we described the two main modules of our system, namely Design Synthesis and Optimization. In Section 5.4, we describe the remaining modules, most of which have been supplied by our industrial sponsors. These modules relate to the analysis routines (solvers) built into the NDA. The chapter concludes with a brief summary.

5.1 The Software System

In this Section, we refer back to Figure 1-2 which shows the objective of the NDA system schematically. The user (or the designer) specifies a set of requirements and constraints

to the system and the system returns a set of solutions, or designs matching the user's requirements. User interaction is required at two stages, once for the design synthesis or design sub-space identification step, and secondly for the optimization step. In this section, first, we will briefly present the anatomy of the software system, and break it down into different parts or modules. Next, we will explain the organization, and show the sequence of steps the user would have to undertake to use this system fully.

5.1.1 Structure

We divide the NDA software system into five modules. In Figure 5-1 below, we have reproduced Figure 2-1, which shows the different modules of the NDA in a schematic fashion.

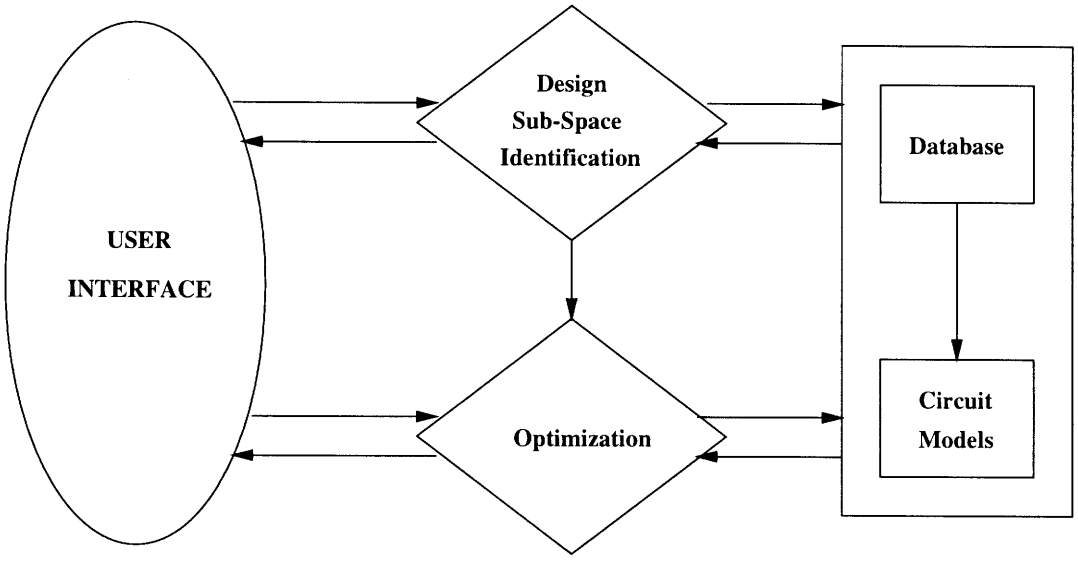


Figure 5-1: The Different Modules of the NDA

The graphical user interface of the NDA (together with pictures of the NDA screen) is described in Section 5.3. Chapter 3 described the Design Sub-Space Identification module, and the Optimization module was described in Chapter 4. The remaining two modules, have been mostly supplied by our industrial sponsors. These modules and their sub-modules (called Rule Sets) are described in Section 5.4.

The user supplies two kinds of inputs to the user interface. First, the user must specify the target limits for Attributes. Attributes are different for every solver and hence, even before specifying attributes, the user has to pick the solver to use (i.e. the kind of machine to design). Attribute Specification is done in the form of target ranges for each of the attributes. The user then is presented with a list of design variables for the solver in question. Constraints on design variables are specified at this point. These constraints take the form of target ranges or limits on the values of the design variables. The user may fix the value of a variable by specifying the same value for the minimum and the maximum limit. The user may also specify an optional step size argument for every variable. Categorical variables are specified by writing down the values of the variables in question. Usually, these variable values are names of supporting parts found in the database. A list of categorical variable values (or choices) may be specified by separating them with commas in the appropriate text field. Some of the inputs are further detailed with the help of diagrams in Section 5.2.

5.1.2 Organization

The NDA system is organized to lead the user through a series of steps in the design procedure. These steps are summarized below.

1. Select the kind of machine to design i.e. select the specific solver to use.
2. Specify target attributes.
3. Specify constraints on design variables. This is optional. Constraints are *not required* for all variables. However, most practical design scenarios will specify a number of constraints on design variables. The user has complete flexibility of leaving a variable completely "open" i.e. not specifying any limits on it, specifying constraints in the form of a range on a variable, or fixing the value of the variable (by specifying a single value for the minimum and maximum limit).
4. Let the system perform the design sub-space identification procedure.
5. View the designs generated during sub-space identification.

6. Specify an attribute to optimize (maximize or minimize). Again this is optional. The user may elect not to execute the optimization step at all. (We must point out at that in our current implementation, we only allow the user to optimize based on one attribute, as required by the TradeOff method. However, the methodology presented in Chapter 4 is sufficiently general, and can allow other objective functions which might be combinations of more than one attribute).

7. Let the system perform the optimization step where it will seek to optimize the specified attribute, while keeping others within the limits specified in Step 2.

8. View the resulting optimized design.

The user interface ensures that the steps are performed in a logical manner. So, absurd operations like optimization before sub-space identification, are not allowed.

5.2 Introduction to Motor Design Variables

Before we proceed to describe the user interface and other modules of the NDA system, it is instructive to outline some of the variables used to describe an induction motor. The anatomy of a generator is largely similar (the rotor geometry is different). This section will also help place the discussion and "on-screen" example of Section 5.3 in context.

Figure 5-2 shows a cutaway view of a single phase induction motor. The figure shown is a CAD solid model, developed by the Advanced Development Center of our corporate sponsors: MagneTek Inc. Induction motors have two major components: rotor and stator. The rotor is mounted on a shaft which rotates and is used to tap the mechanical power output. The shaft is supported by bearings on both sides. Aluminum end rings may be found on both sides of the rotor (for cast and fabricated rotors). The rotor fits inside the stator, leaving a small (cylindrical) air gap. Air gaps are very small, typically a few thousandths of an inch (usually in the range of 0.015 to 0.05 in). The stator is stationary and is attached to the frame. The frame is reasonably longer than the stator and rotor (as shown in Figure 5-2), and its base is usually mounted on the ground with the help of bolts.

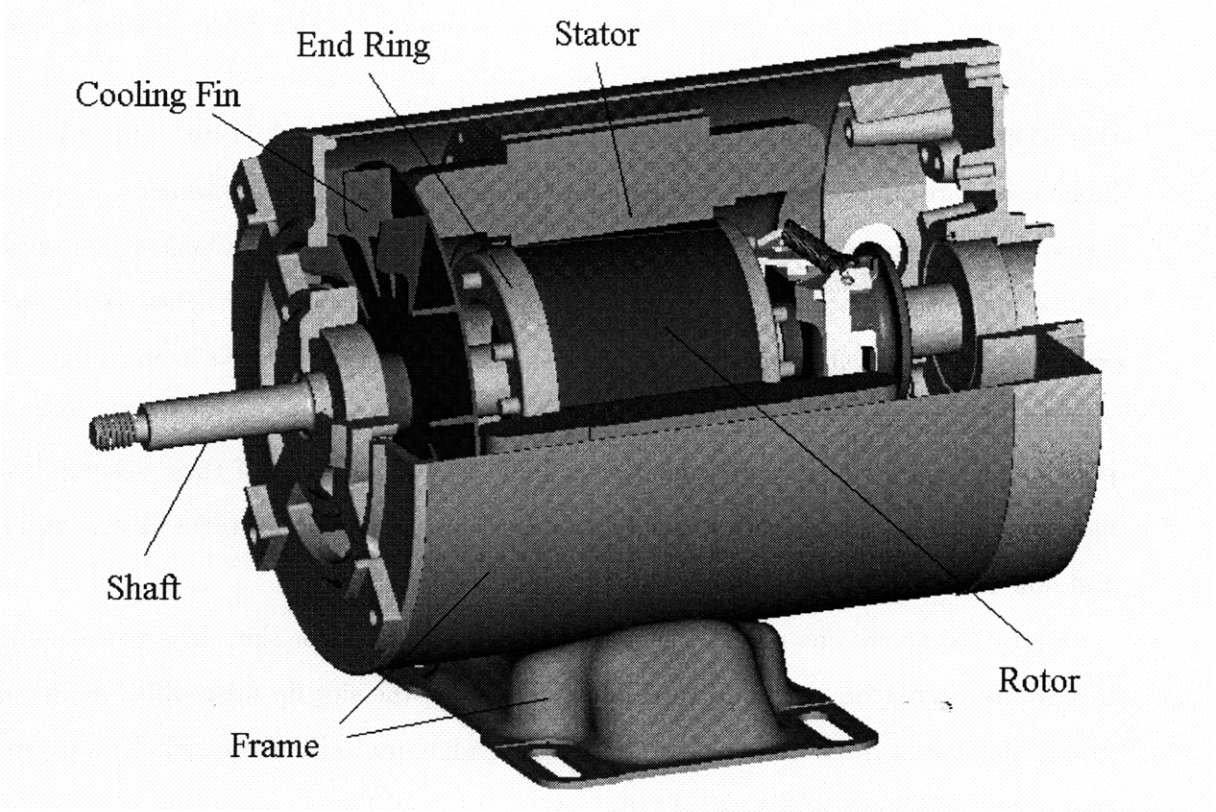


Figure 5-2: Cutaway View of a Single Phase Induction Motor. (Courtesy: *MagneTek Inc. Advanced Development Center, St. Louis*)

Rotors and stators are made of steel, and are usually punched in the form of thin laminations (with a thickness to the order of $0.02in$). A number of these laminations (all insulated from each other) are stacked together to make up the rotor or the stator. Rotors and stators are manufactured in this manner to avoid heavy eddy current losses. After laminations have been punched, slots are then punched or "nibbled" out of these laminations. Figure 5-3 shows a schematic of a stator lamination, with four slots. Most stators would have a much larger number of slots though (In the two polyphase examples presented in Chapter 6 we will see motors with 60 and 36 stator slots respectively). In this schematic, we choose to show only four slots for the sake of clarity. After the stator laminations are stacked, bundles of wires are pushed into these slots through slot openings. These wires are connected to the electrical supply (single-phase or poly-phase), and serve to drive a magnetic flux through the steel and the air gap, into the rotor. These wires are the terminals for electrical power input, and hence in effect, serve to "drive" the motor. These bundles of wires are wound through the slots in a very specific pattern – the two most popular styles are called "lap" and "pyramidal" windings. Both of these schemes would have their own associated set of variables to describe the number of turns of wire in a coil, and how the bundles (coils) are nested through the slots.

Figure 5-3 shows one possible geometrical shape for a stator slot. It is not uncommon to have round bottom or round top slots. Frequently, the slot lip (also called depression), would also have a trapezoidal shaped extension leading up to the main slot. This extension (not shown in Figure 5-3) is called "rise".

Figure 5-4 shows a similar schematic for the rotor lamination. Rotor laminations also have slots which may or may not be similar in shape to the slots on the stator lamination. Round top and round bottom slots are common. The number of rotor slots is also different from the number of stator slots (6 rotor slots are shown in our schematic – we will see examples with 48 and 44 slots in Chapter 6. In contrast to the stator slot, the rotor slot may not always have an opening. In the closed slot case, the slot is a closed "hole" in the rotor lamination. This is feasible since there are no wires to be pushed into the rotor slots (unless it is a wound rotor, in which case, it is wound like the stator). Typically, rotors

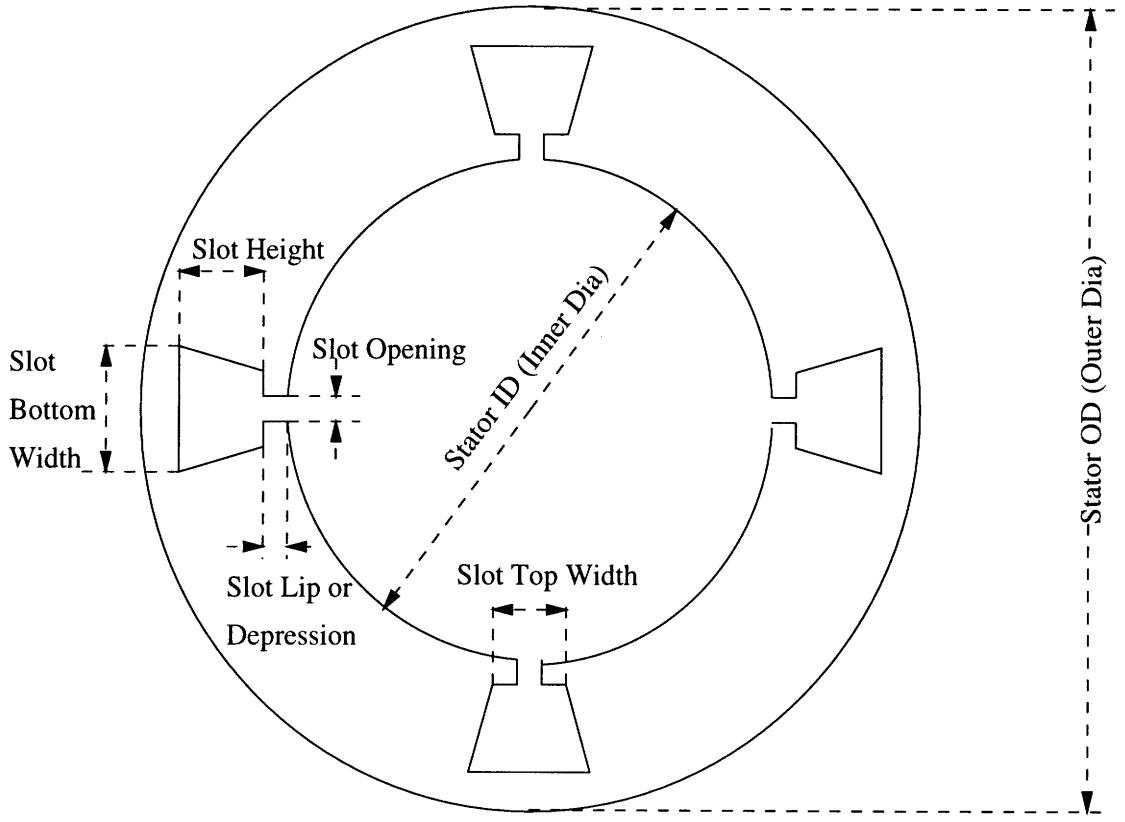


Figure 5-3: Stator Lamination Schematic

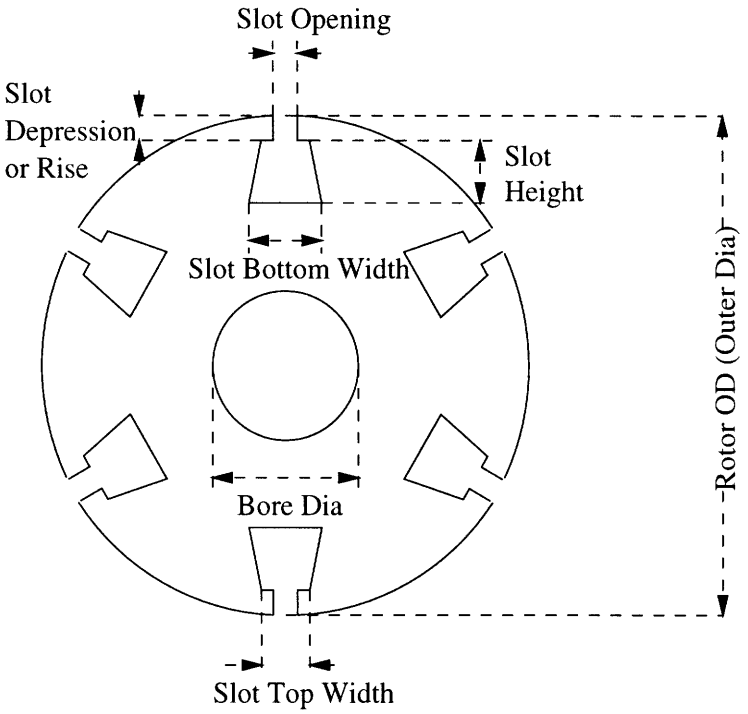


Figure 5-4: Rotor Lamination Schematic

would be cast or fabricated. For cast rotors, aluminum would be cast through all the slots after the laminations have been stacked. At either end of the rotor stack, we would have aluminum end rings (which serve to "collect" the current flowing through the aluminum). For fabricated rotors, aluminum is not cast, but pre-fabricated bars of aluminum are pushed axially through all the rotor slots. This scheme would also have end rings which would serve a similar purpose.

For polyphase motors, we sometimes have double or triple "cage" rotor slots. In that case, the rotor lamination would need additional variables to describe the slots. A multiple cage slot configuration typically looks like a slot of the type shown in Figure 5-4, with another "slot" attached to the bottom of this slot.

The shaft runs through the bore of the rotor laminations.

It may be noted that the rotor has no direct electrical connection. The current flowing through the aluminum is all "induced" by the magnetic flux through the air gap (and hence the name "Induction Motor").

Figure 5-2 also shows some cooling fins on the left hand side. It is common for a motor to have some cooling mechanisms of this kind. These fins may be separate, or they may be attached to end rings, or there may actually be a fan to blow air axially over the motor. Motors with fans are typically called TEFC (Totally Enclosed Fan Cooled) motors.

The design variables used to describe a motor consist of all the variables used to describe all the major components outlined above. The actual number and the actual variables used depend on the sophistication of the electrical circuit model (solver) used to analyze a motor design.

In the following section, we will describe a user interface for the NDA which allows the user to specify constraints on all the design variables, and specify requirements for all the performance variables.

5.3 Graphical User Interface

[96] describes a user interface for the NDA. In this thesis, the NDA has a new user interface which has been totally revamped from the one described in [96]. The earlier user interface

had a different screen layout and was limited only to the design of polyphase motors. It did not have a facility for optimization (which was added as a part of the current thesis project), and it did not allow us to specify stepsize increments for each of the design variables. However, the basic principles remain the same. The user interface is still based on X-Motif and guides the user through the sequence of steps for designing motors and generators. The reader is referred to [96] for a description of X-Motif basics used for designing the user interface. [121] provides a good introduction to X-Window programming in Motif. We also acknowledge the programming manuals for X-Motif programming (Programmer's reference [37], Programming Guide [36], and Style Guide [38]) which were used extensively for designing the user interface.

5.3.1 Screen Layout and Working Screens

The NDA user interface has a menu bar on top and the rest of the screen is available as a work area (with scroll bars where necessary). At different stages of the design process, different "screens" occupy this work area. The NDA starts off with a selection screen where the user is prompted to pick a solver to use (signifying the kind of machine to be designed). The opening screen is shown in Figure 5-5.

There are basically four kinds of "screens" which we would refer to in the rest of this section:

- Attribute Screen – for accepting inputs from the user on target attribute limits.
- Input Screen – there are two screens in this category. Both are used for accepting inputs from the user about constraints on design variables. Design variables presented in the second screen are dependent on certain choices made by the user in the first input screen.
- Output Screen – this screen has only one format but there are two situations where it is employed. As the name suggests, this screen is used for display purposes, and is useful for displaying outputs from the NDA. Firstly, the output screen is used for displaying results from the sub-space identification step. At this stage, certain push-

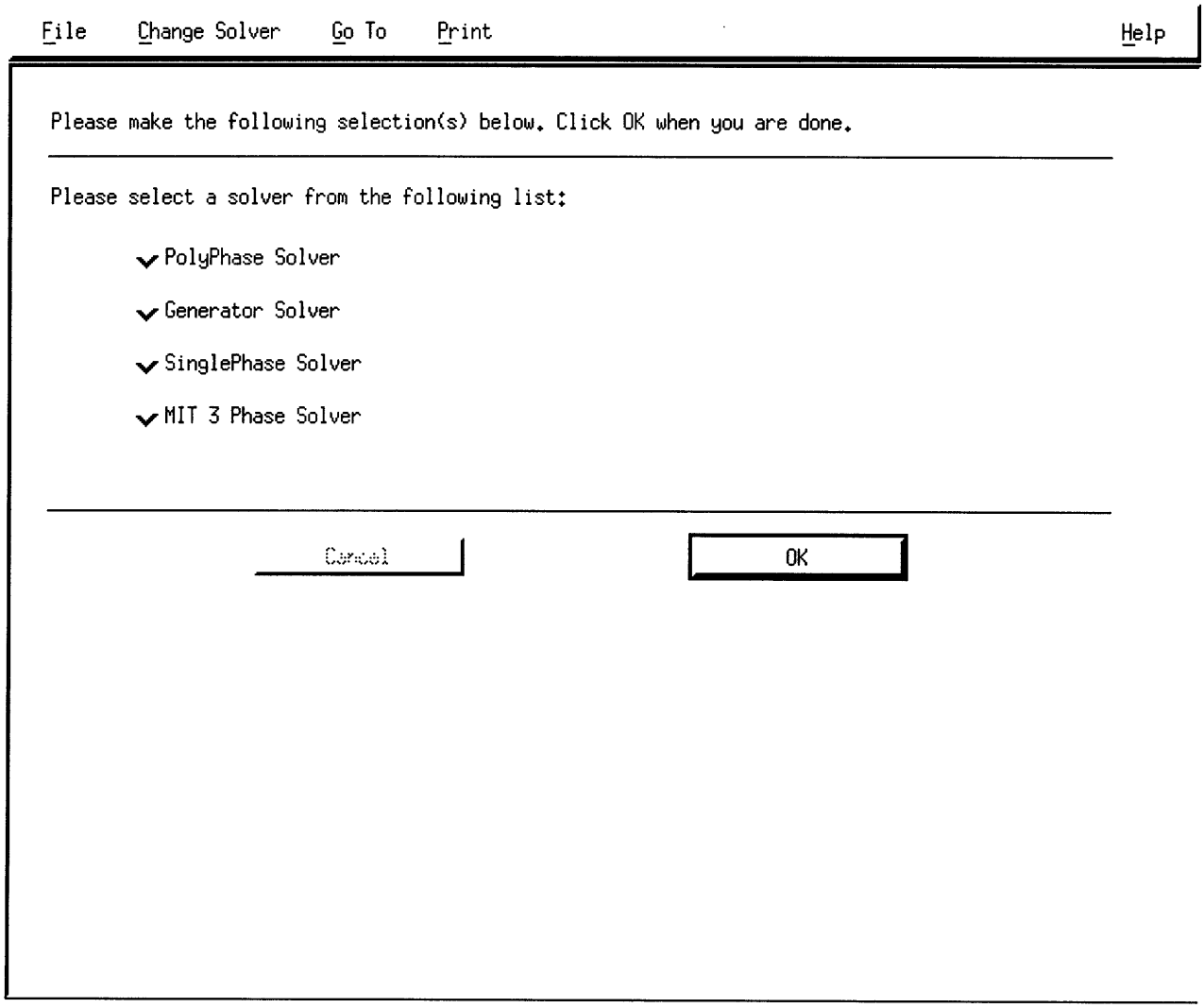


Figure 5-5: The Initial Screen of the NDA

buttons at the top of the screen are active which let the user navigate through the list of generated solutions. The second juncture at which this screen is useful, is when the optimized design has to be displayed. No navigation buttons are active at this stage, however.

- Selection Screens – There are two selection screens in the NDA. The NDA launches with the first one (Figure 5-5) and prompts the user to select the solver to use. The second selection screen occurs between the design sub-space identification step and the optimization step where the NDA prompts the user to select the primary attribute for optimization.

The NDA's menus serve to provide certain features like saving or printing information, and quitting from the application. They also let the user navigate through the different screens at any time. The Menu system has built in Menu accelerators and Mnemonics for keyboard activation of menu items (as alternatives for mouse interaction).

5.3.2 Example Run with the GUI

Perhaps the best way to show the working of the NDA and its user interface, is to present an example and present all the steps and the screens necessary for completing the example. For our example, we select the MIT solver (we cannot choose the other solvers for proprietary reasons), and try to design and optimize a 3 Hp (2238 W) three phase motor. To choose the MIT solver, the user clicks the "MIT 3 Phase Solver" button shown in Figure 5-5 and clicks "OK".

The system then presents the Attribute screen to the user. The Attribute Screen together with the user's inputs is shown in Figure 5-6.

For the purposes of our example, we have selected attribute ranges of: Efficiency ($85 - 100\%$), Power Factor ($75 - 100\%$), Iron Mass ($0 - 20Kg$), and Copper Mass ($0 - 4Kg$). Clicking "Done" presents the user with the First input screen, shown in Figure 5-7.

There are three kinds of variables in the Input Screen: numerical variables, "name" variables, and option variables. The "Design No" field in Figure 5-7 is an example of a

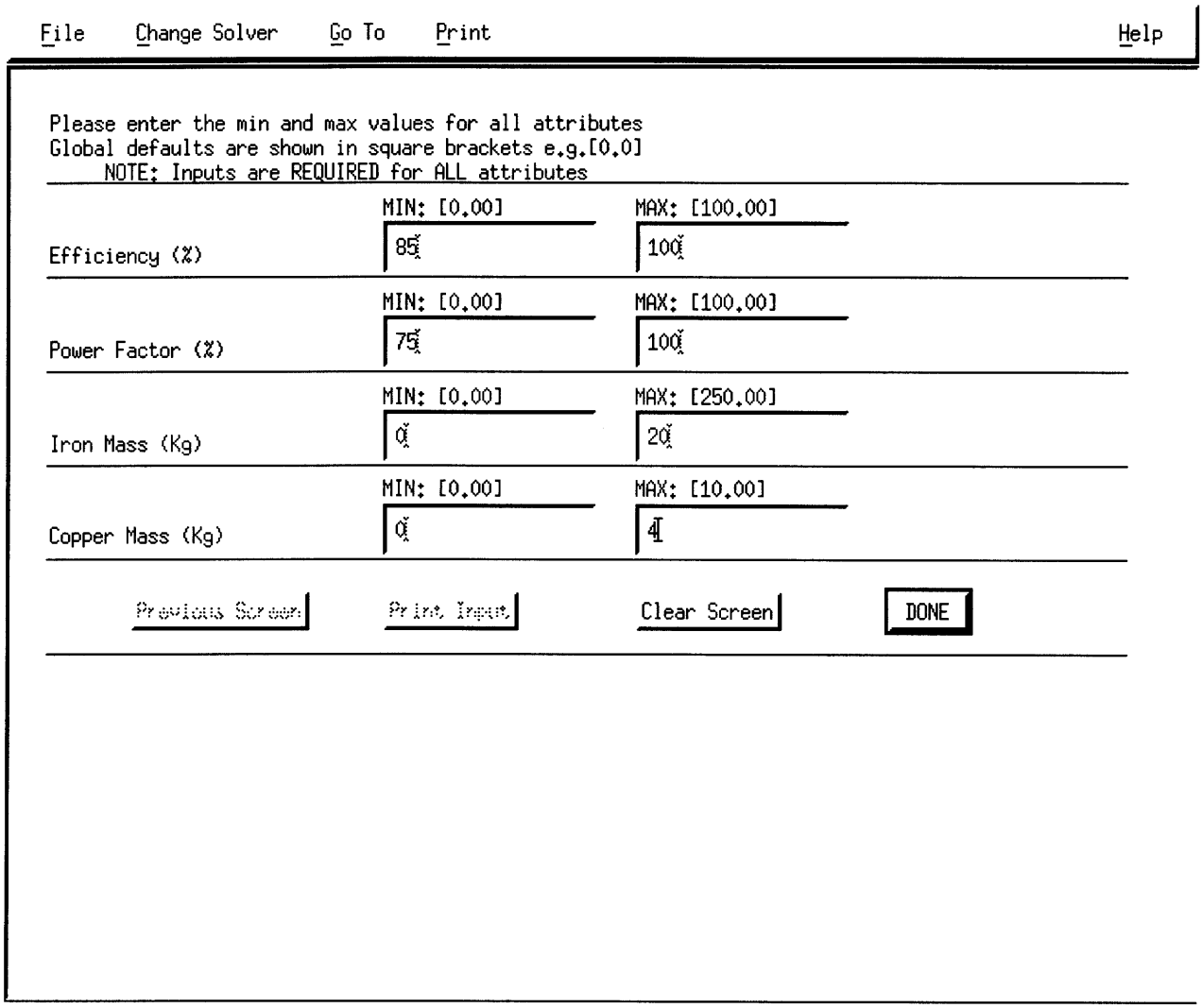


Figure 5-6: The Attribute Screen of the NDA

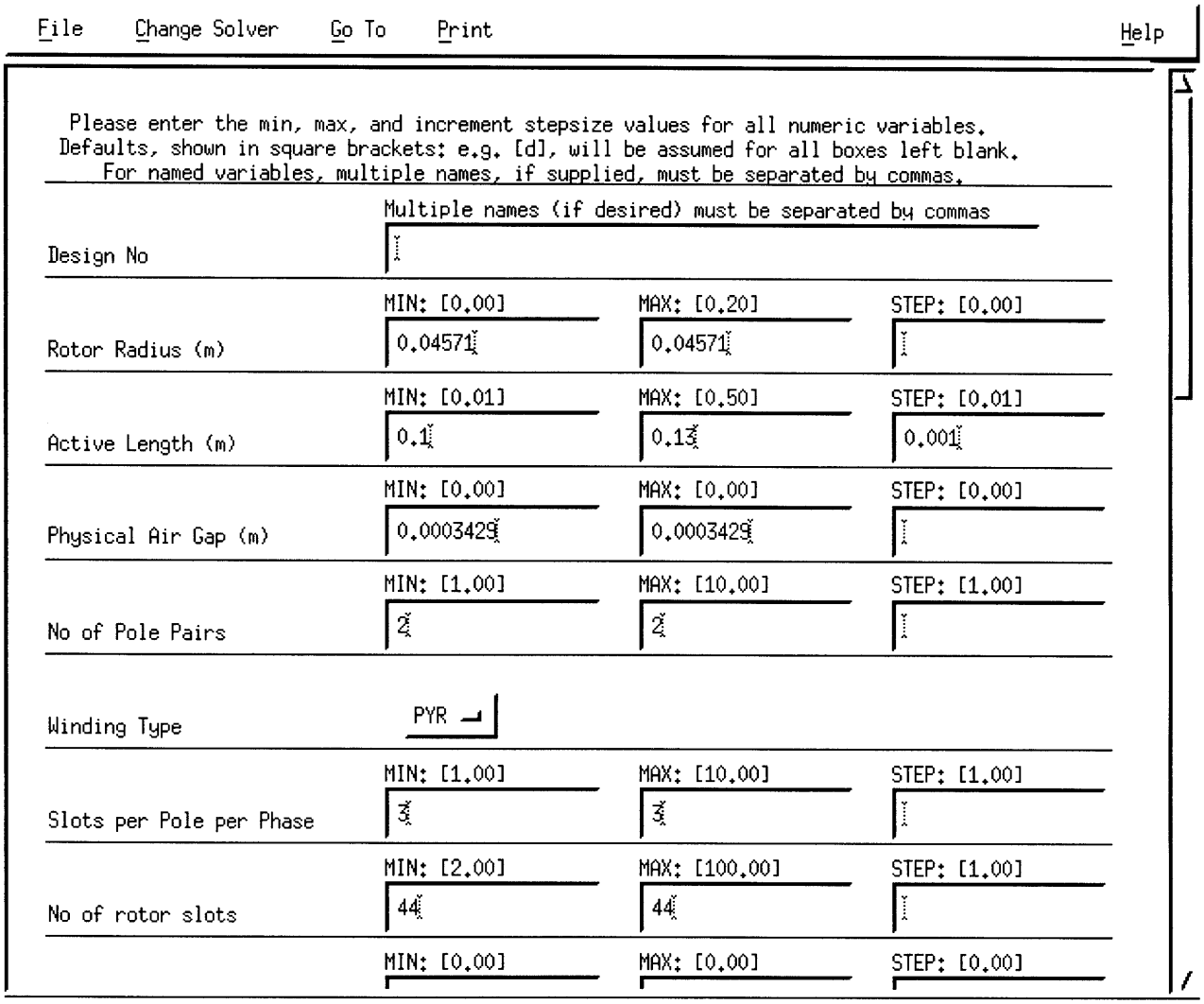


Figure 5-7: The First Input Screen of the NDA

"name" variable where the user is expected to enter a name, if desired. The "Winding Type" field is an example of an option variable. There are only two possible choices: LAP or PYR, and in the example, we have chosen PYR. The most common type of variable is the numerical variable. The user may specify a minimum, maximum, and a stepsize value for any numerical variable. It must be stressed, however, that the user has complete flexibility to specify only the values he desires. For all boxes left blank, default values would be assumed. Default values for all boxes are shown on top of every box. The user has complete flexibility of allowing a variable to vary within a specified range (e.g. Active Length in Figure 5-7), or to fix the value of a variable by providing identical values for the minimum and maximum limits (e.g. Rotor Radius in Figure 5-7).

Since the list of variables is large, we need to scroll down to specify additional variable ranges. For the sake of completeness of the example, this is shown in Figure 5-8 and Figure 5-9.

After the user clicks "Done" in the screen of Figure 5-9, the second input screen is presented (Figure 5-10). It may be noted that the variables in this screen are all related to "PYR" windings. This was one of the choices we made in the First Screen. Had we chosen "LAP" instead, this screen would have been different.

Clicking "Done" on the second input screen starts the design sub-space identification process. This process may take some time to complete, depending on the complexity of the requested design. In our case, we have restricted almost all the variables. Hence, this process completes in about 15 min on an HP 9000 817S. This process may take up to 2-7 hours depending on the type of solver selected and the complexity of the design process. Upon completion, the system informs the user about the number of designs generated (Figure 5-11). The user can then go through and view all the generated designs. One such design is shown in Figure 5-12.

After viewing the generated designs, the user may elect to perform the optimization step. Clicking the "Optimize" button places another selection screen in front of the user. Let us say, we select to optimize on Efficiency (Figure 5-13).

The NDA then starts performing the optimization process described in Chapter 4. This,

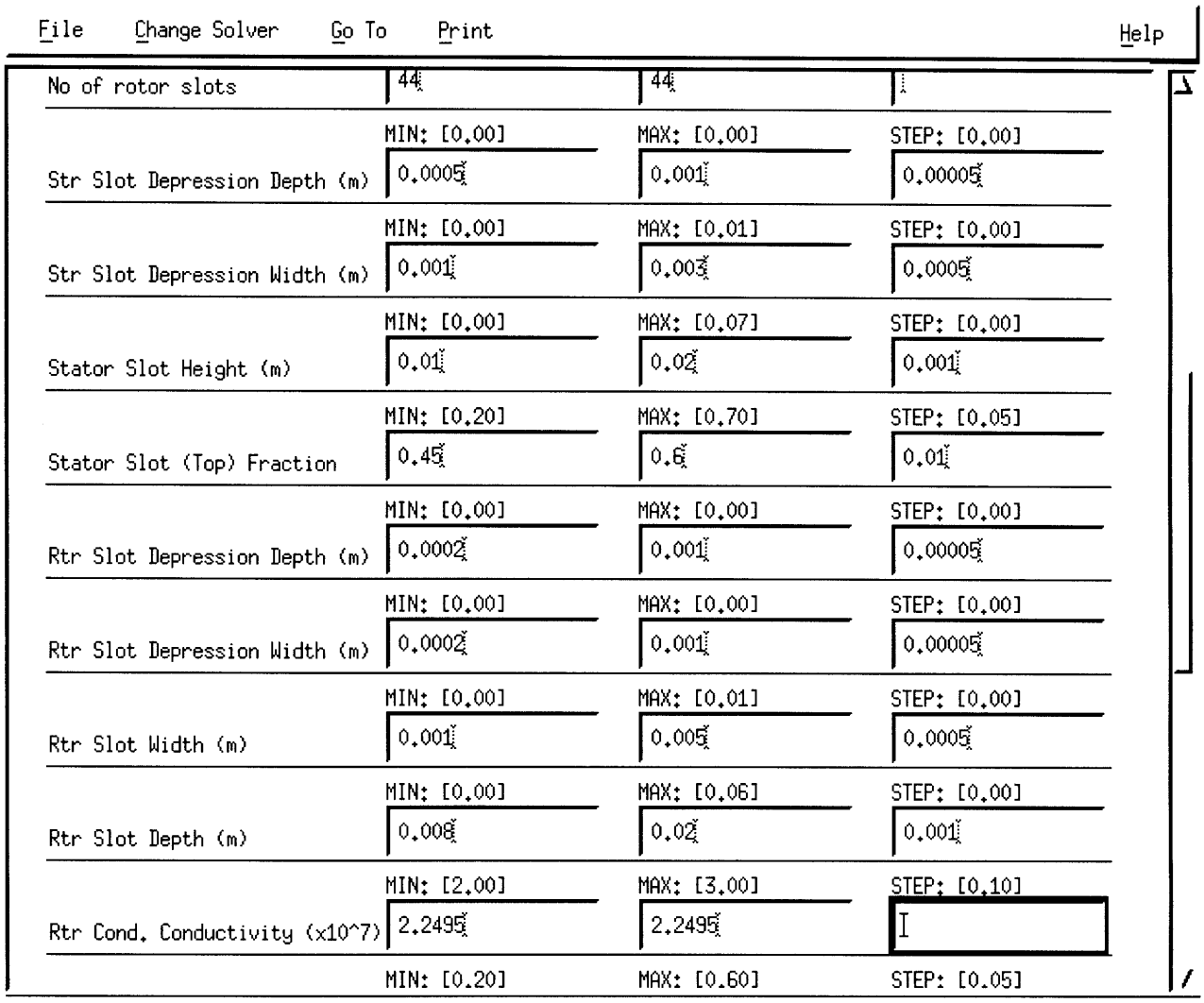


Figure 5-8: The First Input Screen of the NDA – II

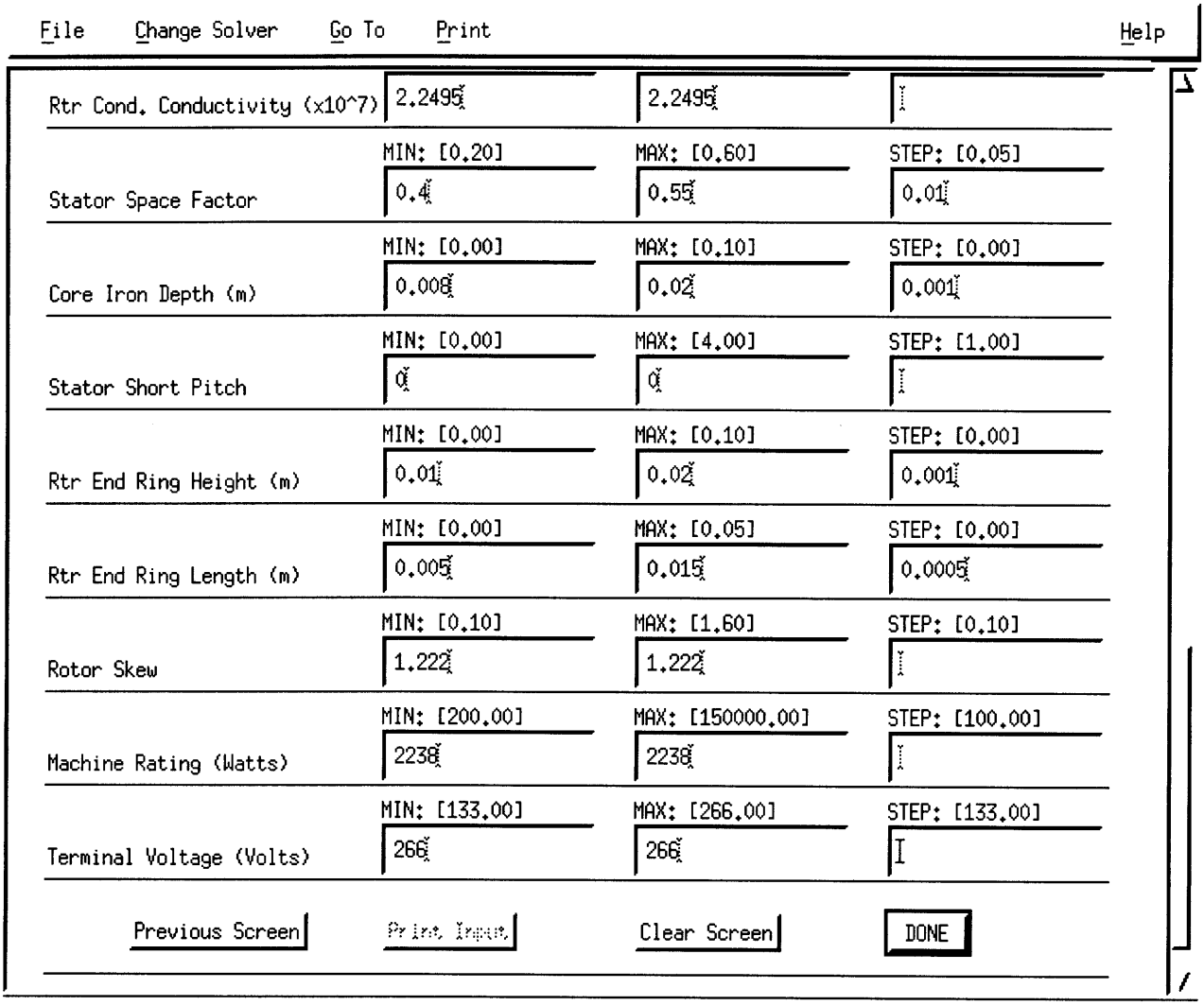


Figure 5-9: The First Input Screen of the NDA – III

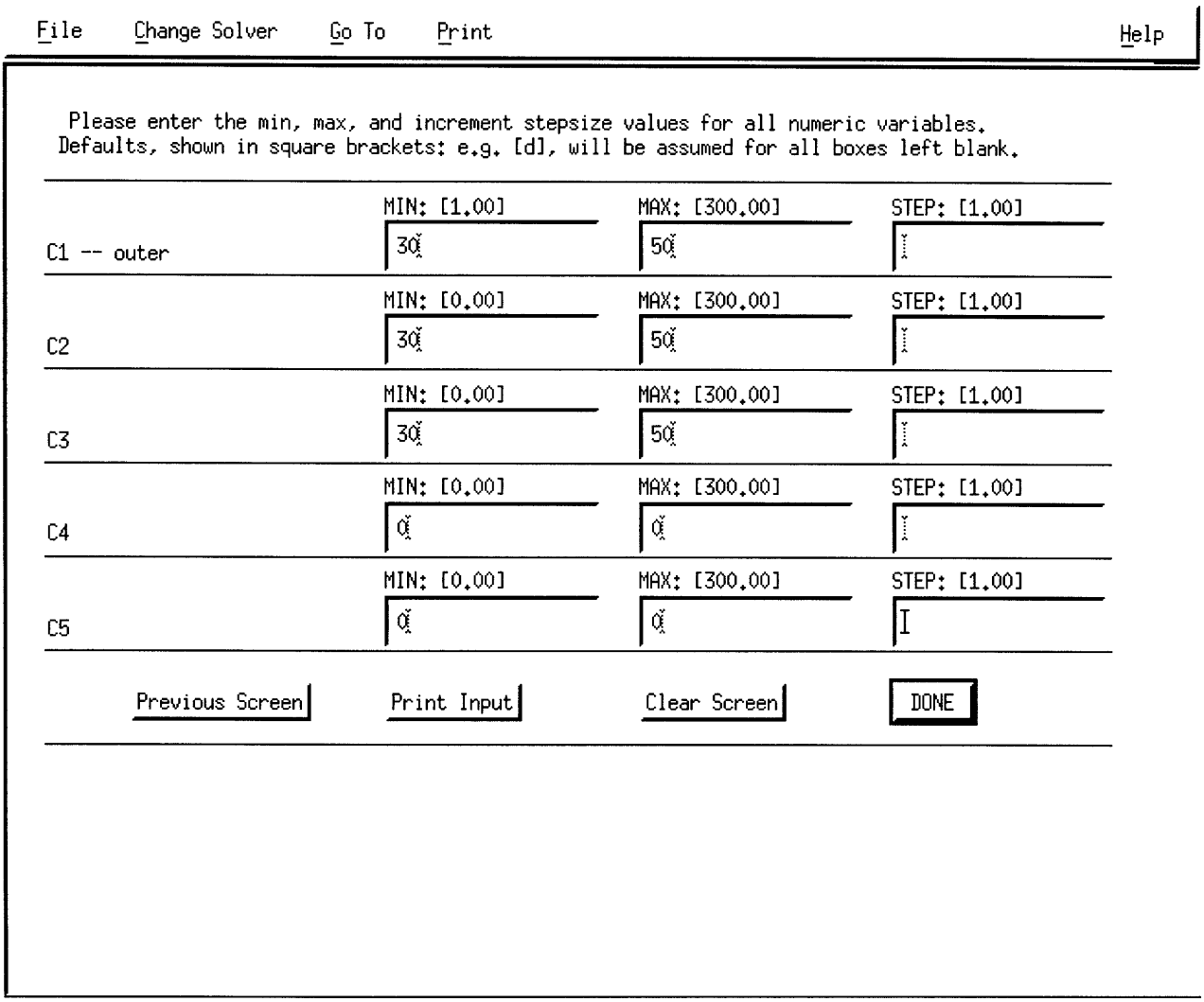


Figure 5-10: The Second Input Screen of the NDA

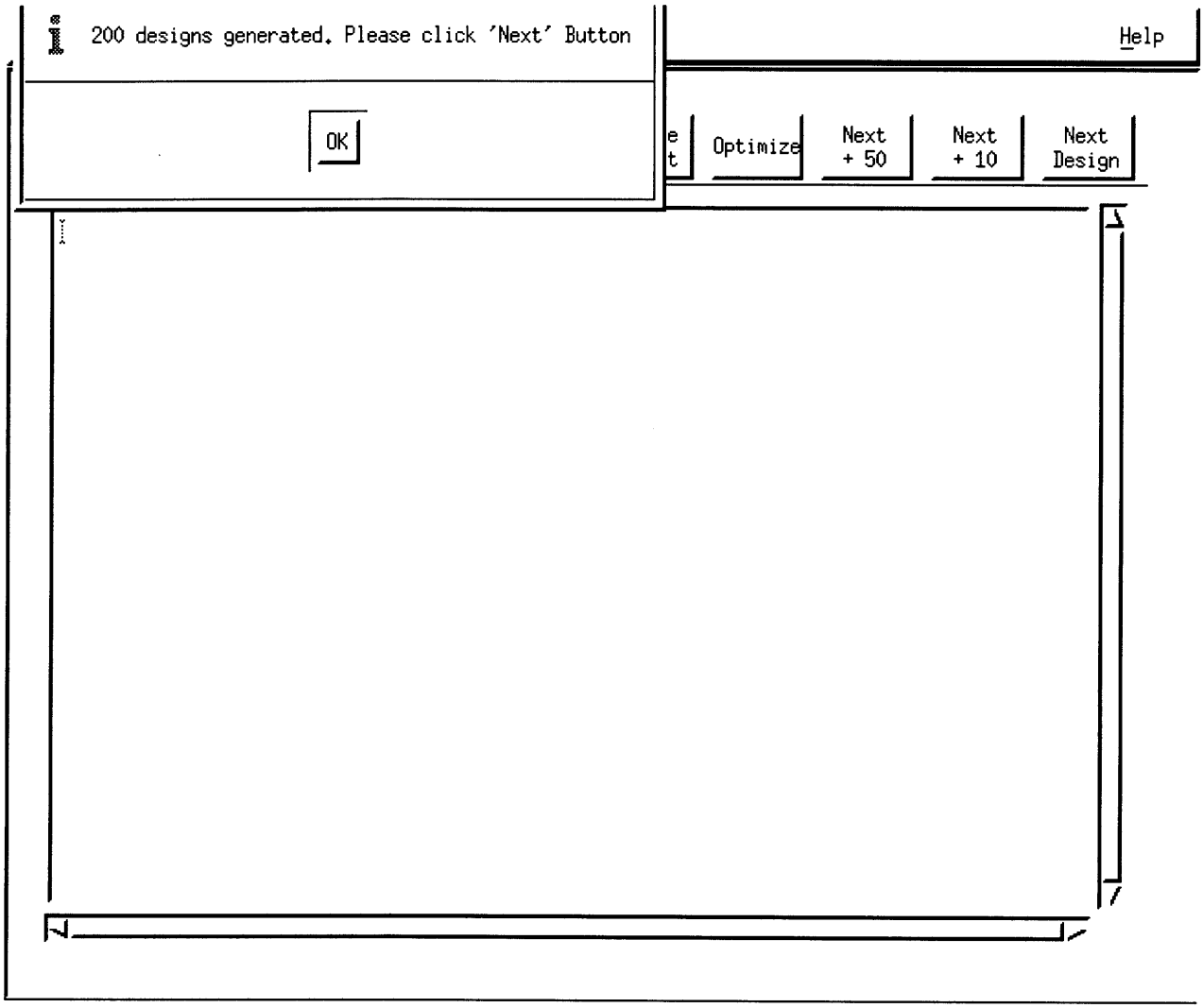


Figure 5-11: Design Sub-Space Identification Complete

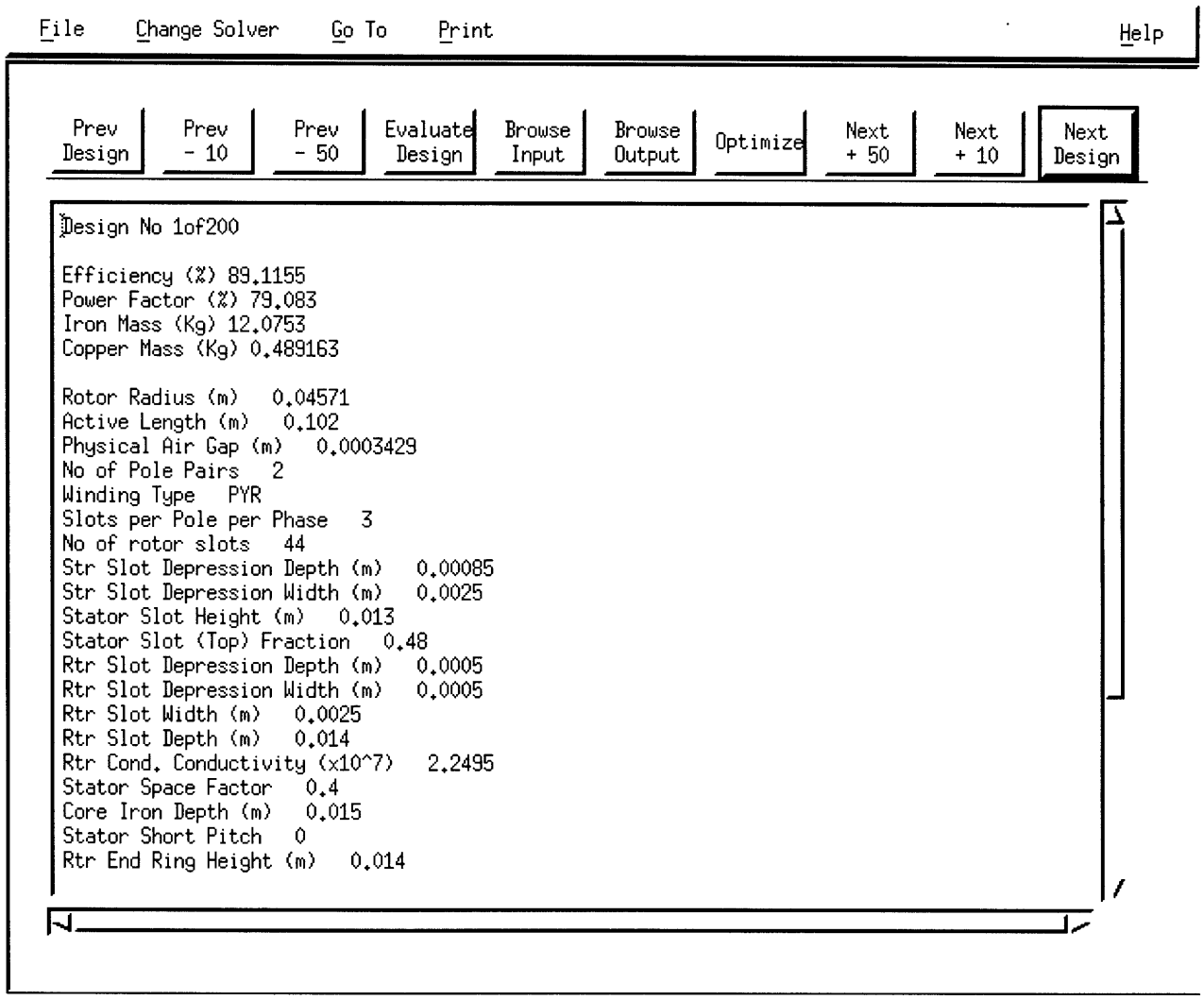


Figure 5-12: One (Sample) Generated Design

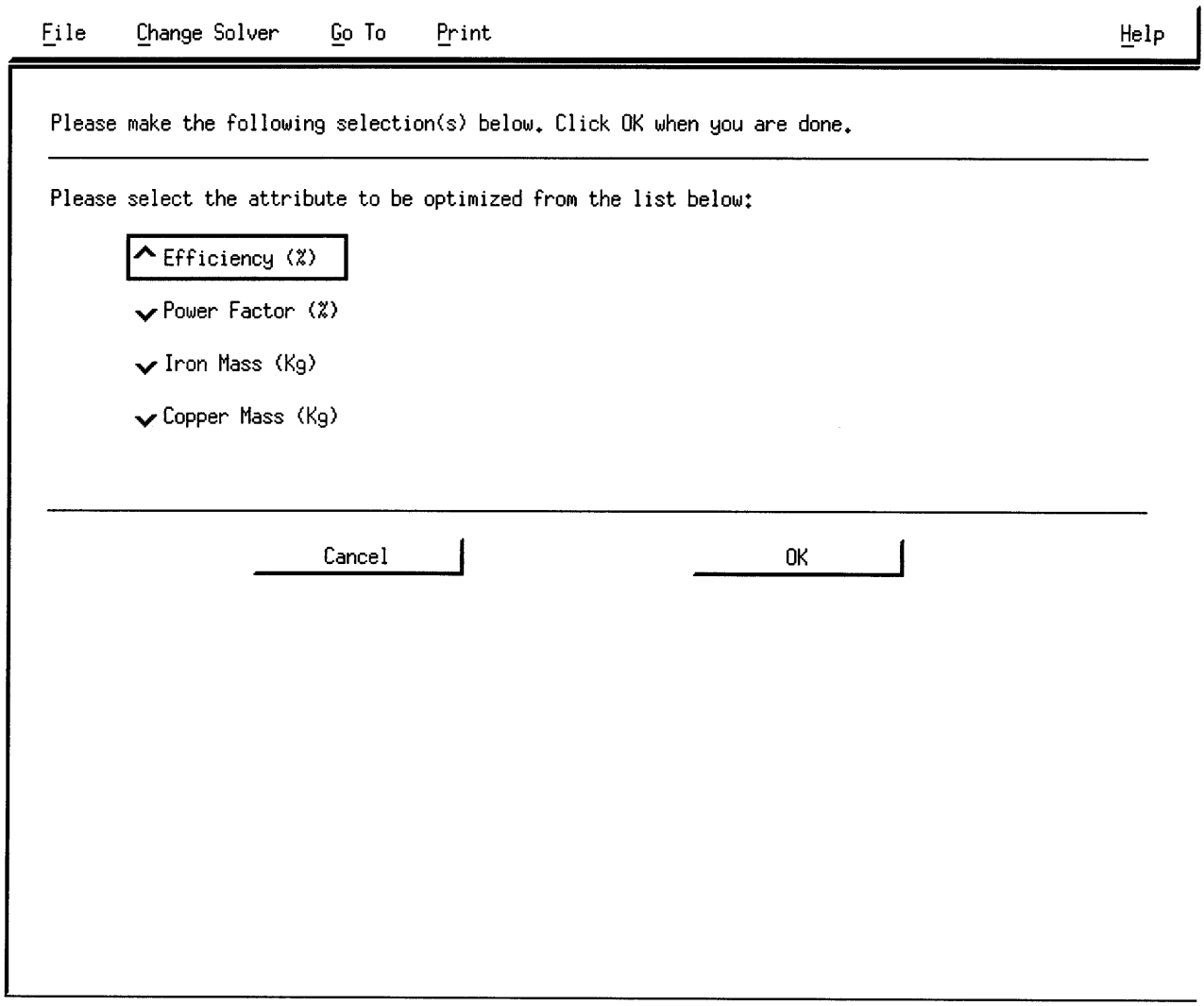


Figure 5-13: Attribute to be Optimized

again, may take a while. On our system, this process completes in another 20 min. Depending on the solver selected and the complexity of the design involved, this process may take upto 7-8 hours. The optimized design is viewed in a manner similar to before (Figure 5-14).

The user may click the "Evaluate" button to view the detailed analysis of this design. The output of this process is the familiar output sheet produced by the solver upon completion of one analysis (Figure 5-15). This feature allows designers in an industrial setting to view outputs in a format they are familiar with, which enhances the "acceptability" of the software as a design tool.

At the end of the optimization step, the natural question is: How did we perform with the optimization process? Let us go back to the starting point for this optimization process. In this case, it is Design No 185of200. This design is shown in Figure 5-16.

A quick comparison tells us that the optimization process improves the efficiency from 89.44% to 91.17%, an improvement of 1.73%. The power factor also increases from 79.76% to 83.87% (up 4.11%). The other two attributes decrease (as might be expected from the TradeOff Method). Iron Mass goes up from $10.36Kg$ to $15.62Kg$ (worse off by $5.26Kg$), and Copper Mass goes up from $0.51Kg$ to $1.19Kg$ (worse off by $0.68Kg$). From the TradeOff Method, one might expect that the Power Factor should also go down (i.e. be worse off). But Power Factor and Efficiency are related quantities and frequently, during optimization processes, an increase in both Efficiency and Power Factor may be observed.

5.4 Solvers and Rule Sets Modules

In this Section, we will discuss some of the components provided by our industrial sponsors, MagneTek Inc. As mentioned in Section 1.2.4 we cannot discuss these components in any detail for proprietary reasons. Here, we will simply mention the different solvers and their rule sets and show some examples of attributes and design variables for every solver. Consequently, the reader may find that our discussion is somewhat sketchy and does not delve into any details. Our aim here is to provide an overview of these components for the sake of completeness, without discussing any proprietary information.

We will begin with a short discussion on rule sets and then go over all the solvers in the

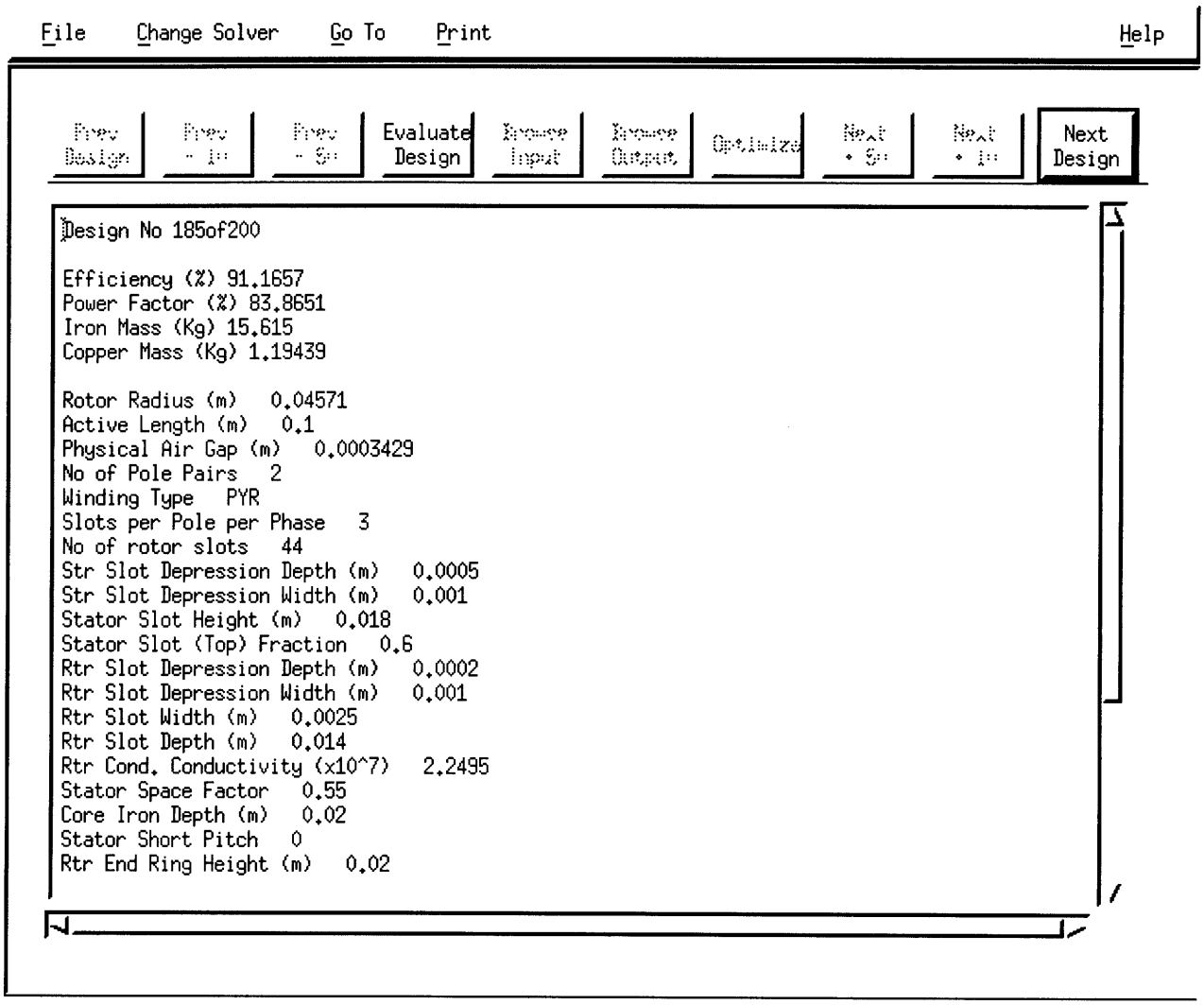


Figure 5-14: The Optimized Design

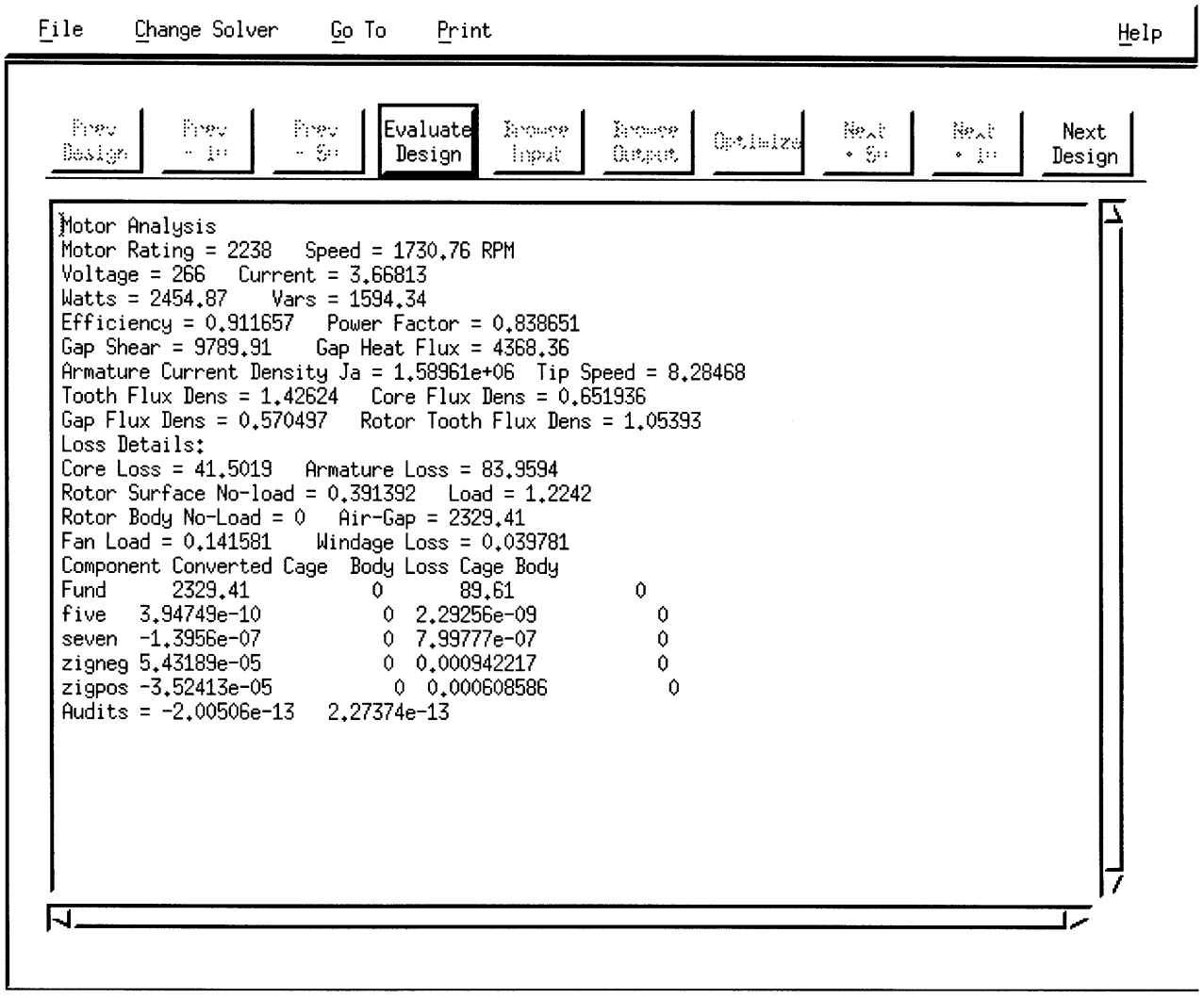


Figure 5-15: Evaluation of Optimized Design

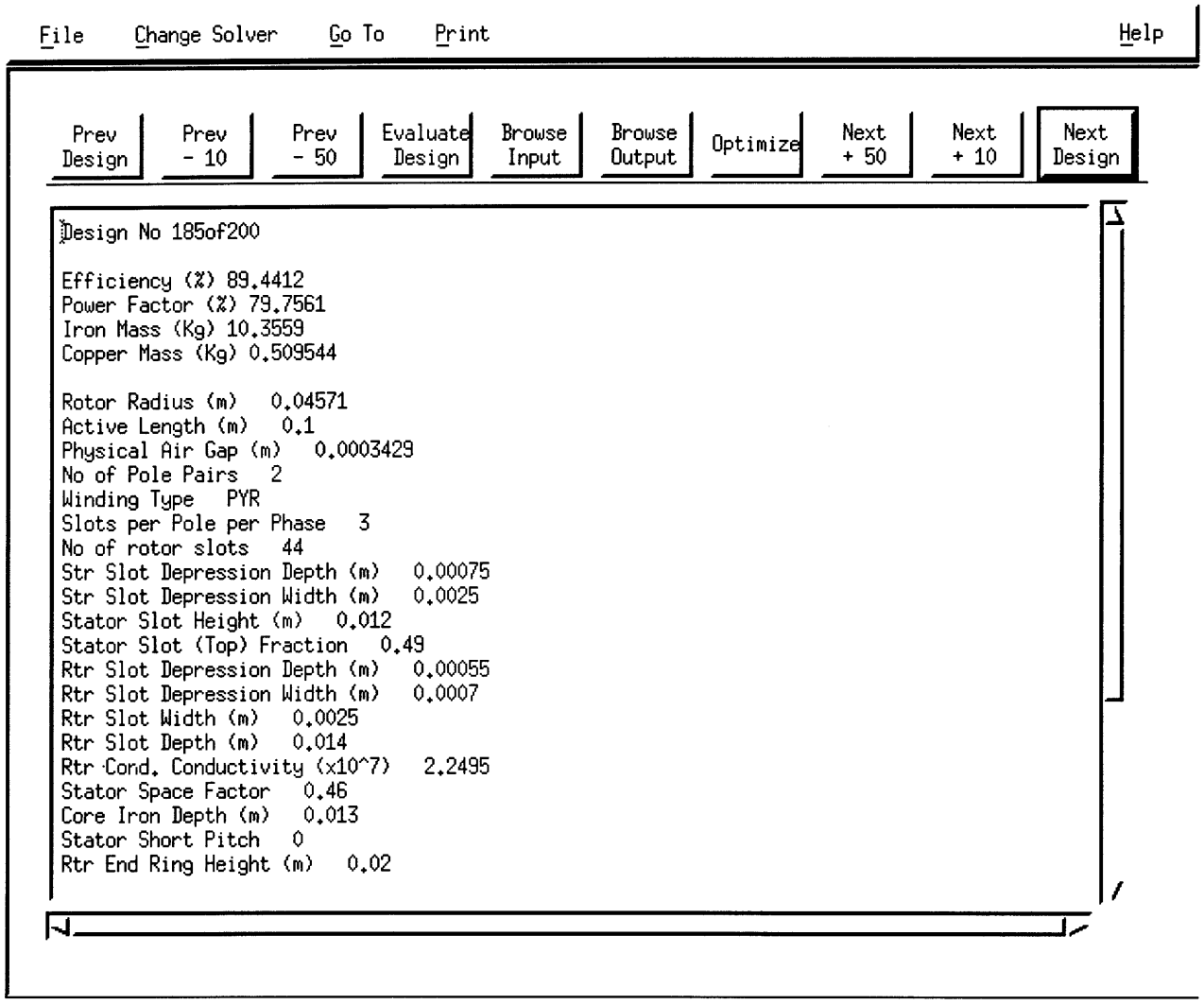


Figure 5-16: The Starting Point which produces the Optimum Efficiency

NDA.

5.4.1 Designing Rule Sets Modules

Rule Sets have been discussed in Section 3.4.2. They are used to perform sanity checks on a design before it is submitted to a solver for analysis. Hence, there are two major places where one encounters rule sets modules in the NDA. One is during the Sub-Space Identification process when every Monte Carlo synthesis trial consists of generating a design at random and submitting it to the solver for analysis. The other is during Optimization when the optimization algorithm decides new sets of design variables to be tried during the "hill-climbing" process. The role of the rule sets module is identical in both scenarios and it is also invoked in an identical manner in both situations.

Presenting infeasible designs to the solvers is inefficient since solvers typically take a long time to analyze a presented design. But on the other hand, having too many "rules" produces designs which are very similar to each other (somewhat like an Expert System), and there is little scope for creativity in arriving at new designs. Hence, there is always a trade-off in designing rule sets modules. The Rule Sets modules rules are hard coded in the program (cannot be modified by the designer), and hence, special care has to be taken to balance this trade-off.

There are two kinds of rules which are required for the NDA. The first relates to increment step sizes of variables. For example, manufacturing constraints might limit core length to vary only in steps of $1/8in$. Such rules or expert guidelines are specified through the user interface. All other remaining rules are specified in the rule sets module.

Rules in rule sets modules may broadly be classified into two types. The first set consists of simple sanity checks like "No negative geometry", "meaningful values of every variable" etc. e.g. No negative geometry might translate to ensuring a positive air gap, and meaningful values might translate to an integer number of slots. These rules are sufficiently general and many of them are common to all the solvers. The second set of rules consists of certain special requirements specific to the solver in question. Examples might include a sinusoidal distribution of windings in stator slots for singlephase motors, and belt angle

Table 5.1: PolyPhase Solver – Attributes and Design Variables.

Attributes	Design Variables
Efficiency	No of Rotor Cages
Power Factor	No of Stator Slots
Breakdown Torque	No of Rotor Slots
Locked Rotor Torque	Rotor Outer Dia
Locked Rotor Current	Rotor Skew
Slot Fill	Phase Belt Angle
Core Volume	Stator Outer Dia
	End Ring Height
	No of Turns
	⋮

dependent slot pitch limits in case of polyphase motors. It is due to this second type of rules that the rule sets modules are very specific to a solver.

5.4.2 Electronic Database

MagneTek's motor design and supporting parts data is maintained in an Oracle database. Supporting parts include slot geometry data, and stator and rotor lamination geometry data, among other things. All solvers query this database to obtain any stored items/data for use during the analysis procedure.

5.4.3 PolyPhase Solver

The polyphase solver, handles two phase and three phase motor designs. Table 5.1 lists attributes of the polyphase solver used by the NDA. Table 5.1 also lists some examples of design parameters used by the solver (in no particular order). The solver's capabilities, however, are not limited to these listed variables.

The Rule Sets related to the polyphase solver are summarized in Table 5.2. It must be re-emphasized that these rules are used in the NDA *before* a design is submitted for analysis. So in a sense, these rules are used to condition the variables before submitting them to the solver. The first five rules in Table 5.2 may be viewed as general "sanity checks". Some of the later ones are specific to the solver. Again, only a brief summary of the entire rule set

Table 5.2: Rule Sets for the PolyPhase Solver

No	Rules and Heuristic Guidelines
1	No negative geometry (positive air gap)
2	Adjust machine diameters (yoke not too deep or too shallow)
3	Ensure that Rotor and Stator Teeth are parallel
4	Adjust slot dimensions (e.g. slot opening < top width)
5	Adjust end rings making sure they fit with machine diameters
6	Ensure all non-relevant variables set to 0 (solver requirement)
7	Slots per pole per phase is an integer
8	Slot pitch limited by slots/pole or slots/phase
9	Parallel paths bound by No of Poles
10	Adjust # of Turns such that slot fill is reasonable

is shown here. The rule sets module, may also use some equations and a set of heuristics to enforce or implement these "rules" or "guidelines". For example, in deciding appropriate (positive) air gaps, an equation is used from [115] to bound the air gap values:

$$gap = 0.005 + \frac{0.0042 \times D_1}{\sqrt{P}} \tag{5.1}$$

where D_1 is the stator Inner Dia and P is the number of poles. The NDA then uses twice this value as an upper bound on air gap.

5.4.4 SinglePhase Solver

The single phase solver is similar to the polyphase solver in its structure. Of course, the singlephase solver handles only single phase motors. The list of supporting parts used in both of these solvers is largely similar. A list of some of the attributes used by the NDA and a short sample of design variables (in no particular order) is shown in Table 5.3.

Similarly, the rule sets modules for the singlephase solver is also largely similar to the polyphase solver. Table 5.4 outlines some of the rules and heuristic guidelines.

Table 5.3: SinglePhase Solver – Attributes and Design Variables.

Attributes	**Design Variables**
Efficiency	No of Rotor Slots
Power Factor	No of Stator Slots
Breakdown Torque	Start Capacitance
Locked Rotor Torque	No of turns (main winding)
Locked Rotor Line Current	No of turns (aux winding)
Max Slot Fill	Rotor Outer Dia
Core Volume	Stator Inner Dia
Capacitor Voltage	End Ring Length
	No of poles
	Stator Outer Dia
	⋮

Table 5.4: Rule Sets for the SinglePhase Solver

No	**Rules and Heuristic Guidelines**
1	No negative geometry (positive air gap)
2	Adjust machine diameters (yoke not too deep or too shallow)
3	Ensure that Rotor and Stator Teeth are parallel
4	Adjust slot dimensions (e.g. slot opening < top width)
5	Adjust end rings making sure they fit with machine diameters
6	Ensure all non-relevant variables set to 0 (solver requirement)
7	Sinusoidal distributions of windings
8	Slot pitch limited by slots/pole
9	Parallel paths bounded by No of Poles
10	Adjust Wire Size such that slot fill is reasonable

Table 5.5: Generator Solver – Attributes and Design Variables.

Attributes	**Design Variables**
Full Load Efficiency	No of Poles
Full Load Field Current	No of Stator Slots
Armature Current Density	Rotor Turns/Pole
Field Current Density	Stator No of turns/Coil
Direct Axis Synchronous Axis	Rotor Winding Height
Direct Axis Transient Sat	Rotor Outer Dia
Stator Slot Fill	Stator Inner Dia
Rotor Slot Fill	Parallel Paths
⋮	Stator Outer Dia
	⋮

5.4.5 Generator Solver

The Generator Solver shares some of the basic similarities with the PolyPhase and SinglePhase solvers but differs in several respects. It uses the same format for inputs and outputs, and queries the database like the other solvers. However, some supporting part geometries and some associated design variables may be different. For example, the rotor lamination and slot, are replaced by a salient pole rotor configuration. Table 5.5 shows a list of attributes and some design variables for the generator solver (again in no particular order).

The rule sets modules for the generator solver is largely similar to the other two solvers. Table 5.6 outlines some of the rules and heuristic guidelines.

5.4.6 MIT 3 Phase Solver

The MIT solver, as the name suggests, has not been supplied by MagneTek. It is derived primarily from the three phase induction motor analysis routine written by Professor Kirtley in 1989 (and subsequently modified in 1995). Some additions are made to this solver to incorporate pyramidal windings. This solver was intended to be an intellectual exercise in writing analysis routines for induction motors. Hence, it is not an "industrial strength" solver. Its performance varies from similar industrial solvers with similar inputs. It usually

Table 5.6: Rule Sets for the Generator Solver

No	Rules and Heuristic Guidelines
1	No negative geometry (positive air gap)
2	Adjust stator slot depths (yoke not too deep or too shallow)
3	Ensure that Stator Teeth are parallel
4	Adjust slot dimensions (e.g. slot opening <= top width)
5	Stator slots per pole per phase is an integer
6	Stator slot pitch bounded by slots/pole
7	Stator winding parallel paths bounded by No of poles
8	Ensure all non-relevant variables set to 0 (solver requirement)
9	Adjust # of Turns such that stator slot fill is reasonable

Table 5.7: MIT 3Phase Solver – Attributes and Design Variables.

Attributes	Design Variables
Efficiency	No of Pole pairs
Power Factor	No of Stator Slots
Iron Mass	Rotor Radius
Copper Mass	Active Length
	Physical Air Gap
	Slots per pole per phase
	Stator slot fraction
	End Ring Length
	End Ring Height
	⋮

predicts slightly higher efficiencies (lower losses). This is primarily due to the fact that this solver allows only rectangular slot geometries, and does not deal with multiple cage configurations. Industrial strength solvers take a long time to develop and perfect, and have the experience of manufacturing real-life motors built into them, in the form of constants and "industrial factors". The MIT solver is fairly new and has not been exposed to a manufacturing environment. It must however be stressed that this solver is fairly reliable, and extremely useful for testing purposes. It was the first solver to be integrated into the NDA and was very useful for testing the methodologies suggested in this thesis.

Table 5.7 shows a list of attributes and some design variables for the MIT solver.

A short list of rule sets or guiding heuristics is displayed in Table 5.8.

Table 5.8: Rule Sets for the MIT 3Phase Solver

No	Rules and Heuristic Guidelines
1	No negative geometry (positive air gap)
2	Adjust machine diameters (yoke not too deep or too shallow)
3	Ensure stator and rotor teeth fractions are reasonable
4	Adjust slot dimensions (e.g. slot opening < top width)
5	Adjust end rings and machine lengths
6	Adjust # of Turns such that air gap flux density is reasonable

5.5 Recapitulation

In this chapter, we considered the NDA from the point of view of the software system. We identified five important parts or modules which constitute the NDA. Two of these modules were covered in Chapters 3 and 4. This chapter describes the remaining three: the user interface module, solvers and database. We begin with an introduction to some of the variables of interest in describing an induction motor. This helps place the user interface module in context. The user interface module is described in some detail with an "on-screen" example in Section 5.3. Solvers and database are components which are supplied by our industrial sponsors. Hence, for proprietary reasons, they cannot be described here in detail. Section 5.4 introduces these modules and provides a brief overview.

Chapter 6

Results and Analysis

So far in this thesis, we have talked about the methodologies used, and the structure and organization of the software implementation. In Chapters 3 and 4, we presented an example with the MIT solver. In this chapter, we will demonstrate the usage of the NDA with examples from an industrial design environment. We will talk about the capabilities of the NDA, and demonstrate how we can use it in conjunction with other company-specific information to yield a powerful and practical decision making tool for a firm. As mentioned before, we will only present summary information from these examples, for proprietary reasons. No actual designs, solver details, or part geometries would be presented or discussed. Similarly, all other analyses requiring company-specific information, would be performed using estimated quantities. Nevertheless, the summary information presented here would be insightful and would go a long way in demonstrating the satisfactory performance of the NDA in an industrial setting.

We begin by presenting a brief summary of results using different solvers (Section 6.1). Several example runs are performed for each of the solvers. In Section 6.2, we discuss the accuracy and significance of the results obtained, using experimental design techniques to estimate the variability in the solvers' predictions. Section 6.3 provides a look at sensitivity analysis on results obtained with the NDA. Section 6.4 touches on the cost implications of the results suggested by the NDA. This is followed by a short section on using the NDA for improved decision making. The chapter concludes with a brief summary.

6.1 Examples and Results

We will devote this section to presenting a comprehensive set of examples and results from the industrial solvers built into the NDA. For every solver, we come up with a series of examples where we progressively increase the complexity of the design process. The first design for every class of machines is simple with most of the geometry held relatively fixed. The winding parameters (Number of turns of wire in a slot etc.) are allowed to vary. With this set of examples, we would expect the least amount of improvement in efficiency (or any other attribute). The next set of designs for every class of machines allows some of the geometry to vary (e.g. machine diameters would be fixed but slot geometries would vary). Finally, we try designs where critical geometric parameters like stator inner diameter (ID) and rotor outer diameter (OD) are allowed to vary.

This line of investigation, where we allow more variables to vary in each subsequent example, represents a systematic line of inquiry for challenging the NDA. When more variables are allowed to vary (especially critical ones), the design process gets progressively more complex. The last set represents the most challenging set of conditions for the NDA. Moreover, this progression of examples makes perfect sense from an industrial perspective. Punches and dies used for stator and rotor laminations are large and expensive pieces of equipment, and hence geometry changes are more expensive to implement. Particularly, the larger dies are more expensive, and hence changes to lamination diameters e.g. stator ID and rotor OD, are the most expensive.

Our aim is to have two runs for every example: one optimizing efficiency and the other optimizing breakdown torque. We will focus on these two attributes for the purposes of our examples and results. These two attributes represent the most important attributes for poly and single phase motors. Moreover, it is difficult to represent information on paper while focusing on more than two attributes at a time. Hence, in the interests of brevity, all examples will only show these two attributes. Other attributes are not shown here, but all of them fall within the user specified limits.

We must also emphasize that all our results here are based on values computed/predicted by the solver. We have effectively assumed that the solver predicts "perfectly" accurate

values. For example, when the NDA algorithms report an increase in efficiency based on the solver reported values, we assume that the design in question does have a better efficiency. We do not consider prediction errors associated with the solver reported values. This assumption is examined and analyzed in greater detail in Section 6.2.

A note of caution about all the examples in this section: the base (starting) design used for each of these examples is an actual design manufactured by our sponsors. Hence, each of the starting points, represents a commercially viable design. Any improvement in these designs would be considered significant. For example, in some of the following results, we see improvements of 0.2% in efficiency. While this number may look pretty small by itself, when we consider that this is an improvement over an existing commercial design, it is fairly impressive. In Section 6.4 we present further analyses to quantify the benefits of this marginal increase in efficiency.

6.1.1 PolyPhase Motors

Two sets of experiments (with two very different motor designs) were performed with the polyphase solver. Table 6.1 shows the two motors and the attribute targets used in our experimentation.

For each of these experiments, three examples were tried. The first example had the most restricted set of variables. Hence, the first example is closest to the base (starting) design. More variables are allowed to vary in the two subsequent examples.

For each example, two runs were performed, one with the aim of maximizing efficiency and the other with the aim of maximizing breakdown torque of the machine. Thus, in all, 12 runs were performed with the polyphase solver.

These runs are labeled as follows: Each label starts with a letter P (for polyphase solver). It is followed by a digit (1 or 2) identifying the experiment (150Hp or 3Hp motor). This is followed by a letter A, B, or C for the three examples. The last letter is E or T representing an Efficiency or Torque run.

Hence, P1CT run implies 150 Hp polyphase motor, example C, and a run which seeks to maximize breakdown torque.

Table 6.1: Two PolyPhase Motors Considered

Specifications	**Motor 1**	**Motor 2**
Hp Rating	150Hp	3Hp
No of Poles	4	4
Target Speed	1770rpm	1750rpm
Frequency	60 Hz	60Hz
Voltage	460 V	460 V
Winding	LAP	PYR
No of rotor cages	3	1
Belt angle	60	120
No of slots (stator/rotor)	60/48	36/44
Temp Rise	100	75
Skew	0.81	1.222
Friction & Windage	700 W	10 W
Attribute Targets:		
Efficiency (%)	90-100 %	85-100 %
Power Factor (%)	75-100 %	70-100 %
Breakdown Torque (lbft)	> 1200	> 40
Locked Rotor Torque (lbft)	> 1100	> 30
Locked Rotor Current (Amps)	< 1500	< 85
Slot Fill (%)	$< 75\%$	$< 75\%$
Core Volume (cubic inches)	< 2400	< 165

Table 6.2: Three Examples for the P1 Motor

Variables to Design	Example A	Example B	Example C
Rotor slot widths, core length, par paths, pitch, wire sizes, #turns, #cond/turn	Yes	Yes	Yes
Stator Slot, End Ring	No	Yes	Yes
Stator ID & Rotor OD	No	No	Yes

Table 6.3: P1 Efficiency Runs

Run ID	BaseEff (%)	Best Generated	Best Optimized	AveImpr (%)	TradeOff (lbft)(%)
P1AE	93.8%	94.0%	94.0%	0.1%	12 (0.9%)
P1BE	93.8%	94.1%	94.2%	0.2%	-2 (-0.1%)
P1CE	93.8%	94.1%	94.2%	0.4%	-30 (-2.19%)

Table 6.2 shows the variables which are varied in each of the three examples for the 150Hp motor (or the P1 series of experiments).

Tables 6.3 and 6.4 show summary results for the efficiency and torque run respectively for the P1 (150Hp) motor.

In these tables, the base value is the value of the base (starting) design. As mentioned before, the base design is a design which is currently manufactured by MagneTek. Best Generated is the best value generated by the MonteCarlo process. Best Optimized is the highest value obtained after the optimization step. Generally, these two values are *NOT* from the same design i.e. the best design after optimization need not come from using the best generated design as its starting point. The AveImpr column reflects the average improvement during the optimization step (between the Monte Carlo and Optimization steps). TradeOff reflects the average effect on the other attribute during the optimization

Table 6.4: P1 BreakdownTorque Runs

Run ID	BaseTrq (lbft)	Best Generated	Best Optimized	AveImpr (lbft)(%)	TradeOff (%)
P1AT	1356	1618	1709	30 (2.2%)	0.0%
P1BT	1356	1593	1786	74 (5.5%)	-0.1%
P1CT	1356	1816	1915	126 (9.3%)	-0.2%

step. e.g. for P1CE, the ave increase in efficiency is 0.4%, and the average change in torque (decrease of 30 lbft or 2.19%) for this optimization is shown in the tradeoff column. All torque values are in lbft and all efficiency values are in %. Torque changes are also shown as percentage deviations from the base value. All efficiency values are shown to the first decimal place and all torque values are rounded off to the nearest whole number.

In the P1 Efficiency runs, compared with the base (starting) design, the slot fills are higher, all rotor slot depths are lower, rotor slots are narrower, and where permissible, stator slots are deeper and narrower. In P1CE, the air gap is slightly larger. The wire weights, and air gap flux densities are also significantly higher. It must be emphasized, however, that these are only general trends. Due to interactions among variables, it is very difficult to clearly identify a set of changes which would improve the efficiency (this is the precise reason why this is a hard problem and we need a tool like the NDA to achieve any significant improvement). This would be true for all subsequent examples also.

In the P1 Torque runs, compared with the base design, the top rotor cage is narrower and the bottom cage is smaller overall. The top cage depth is lower (where permissible), and end rings are smaller, on the whole. Air gap flux densities are similar to P1 Efficiency runs but the wire weights are comparable to the base design. Consequently, slot fills are lower than in the base design.

The examples (and the variables allowed to vary) for the P2 series or the 3Hp polyphase motor, are shown in Table 6.5.

Tables 6.6 and 6.7 show the summary results for the efficiency and torque runs with the

Table 6.5: Three Examples for the P2 Motor

Variables to Design	Example A	Example B	Example C
Core Length, wire size, #cond, #cond/turn, rotor & stator slots	Yes	Yes	Yes
End Ring	No	Yes	Yes
Stator ID & Rotor OD	No	No	Yes

Table 6.6: P2 Efficiency Runs

Run ID	BaseEff (%)	Best Generated	Best Optimized	AveImpr (%)	TradeOff (lbft)(%)
P2AE	85.9%	87.6%	87.8%	0.5%	-1 (-2.3%)
P2BE	85.9%	88.0%	88.0%	0.5%	-1 (-2.3%)
P2CE	85.9%	87.8%	88.0%	0.5%	-1 (-2.3%)

P2 (3Hp) polyphase motor. These tables are analogous to Tables 6.3 and 6.4 respectively.

For the P2 series of runs, as opposed to the P1 series, the air gap flux densities are always lower than the base design, and wire weights are always higher. In the P2 Efficiency runs, the slot fill is larger, end ring heights are smaller, end ring lengths are larger, stator slot depths are smaller, and rotor slot widths are larger. Air gap in P2CE and P2CT are similar to the base design.

By contrast, in the P2 Torque runs, both the stator and rotor slots are wider, and the rotor slot depths are smaller, but the stator slot depths are usually larger. End rings are again smaller.

It should be noted that in both the P2E and P2T runs, we see very clear trade-offs between efficiency and torque. This clearly shows that we are on the Efficiency-Breakdown Torque pareto frontier.

The P1 and P2 series motors make an interesting comparison. Efficiency improvements

Table 6.7: P2 BreakdownTorque Runs

Run ID	BaseTrq (lbft)	Best Generated	Best Optimized	AveImpr (lbft)(%)	TradeOff (%)
P2AT	43	49	52	4 (9.3%)	-0.3%
P2BT	43	50	54	5 (11.6%)	-0.4%
P2CT	43	50	54	5 (11.6%)	-0.6%

are much larger in the P2 series compared to the P1 series motors. There are two possible reasons for this.

- The P1 motors are large and therefore expensive to build. Hence, their design is always well honed and carefully optimized by hand before an (expensive) prototype is built. The P2 motors on the other hand, are much smaller and much less expensive. Hence, special care is not needed to squeeze the maximum efficiency out of this motor.

- The P2 motor is for a (specific) compressor application. It is in a sense, a special purpose motor, where the final speed is required to be within tight limits. This restricts the set of choices available to a designer. For the purposes of the NDA however, speed is not an attribute, and hence it is effectively ignored while trying to derive the best possible efficiency from this basic configuration. Hence, there is potentially more scope for improving efficiency.

6.1.2 SinglePhase Motors

Experimentation for the SinglePhase solver follows exactly the same pattern as that for the polyphase solver shown in the above subsection. Again, we choose two classes of machines and perform two series of experiments. In every experiment, we have three examples with increasing levels of complexity. Each example is run twice, once for maximizing efficiency, and once for maximizing breakdown torque. These 12 runs are labeled similarly. Each label starts with a letter S (signifying the singlephase solver). It is followed by 1 or 2 for the

Table 6.8: Two SinglePhase Motors Considered

Specifications	**Motor1 (S1)**	**Motor2 (S2)**
Hp Rating	0.75Hp	3Hp
No of Poles	2	4
Target Speed	2850rpm	1730rpm
Frequency	50 Hz	60Hz
Voltage	240 V	115 V
No of slots (stator/rotor)	18/13	36/48
Temp Rise	75	80
Skew	0.72	1.5
Friction & Windage	40 W	46 W
Type	Split Capacitor	Capacitor Start
Attribute Targets:		
Efficiency (%)	60-80 %	75-100 %
Power Factor (%)	80-100 %	75-100 %
Breakdown Torque (ozft)	> 35	> 300
Locked Rotor Torque (ozft)	> 7.5	> 400
Locked Rotor Current (Amps)	< 17	< 230
Slot Fill (%)	$< 70\%$	$< 70\%$
Core Volume (cubic inches)	40-100	< 240
Capacitor Voltage (V)	230-370 V	0-150 V

experiment or the motor class. This is followed by A, B, or C corresponding to the three examples, and ends with an E or T for the efficiency or the torque runs.

The two classes of singlephase motors considered are shown in Table 6.8, together with their respective attribute targets.

Unlike the P1 and P2 series, the examples are identical for the S1 and S2 series i.e. the same set of variables is varied in S1B and S2B, and so on. The variables to be varied in every example are shown in a tabular form in Table 6.9.

Results from the S1 series of experiments are shown in Table 6.10, and Table 6.11, for the efficiency and breakdown torque maximization runs respectively. Similarly, results from the S2 series of experiments are shown in Table 6.12 and Table 6.13.

In the S1 series Efficiency runs, the air gap flux densities are lower than the base design, whereas they are higher in the Torque runs. Full Load Capacitor Voltages are also higher (close to the max limit) in the torque runs.

Table 6.9: Three Examples for SinglePhase Motors

Variables to Design	Example A	Example B	Example C
End Ring Length, core length, par paths, pitch, wire sizes, #turns, #cond/turn	Yes	Yes	Yes
Slots, End Ring	No	Yes	Yes
Stator ID & Rotor OD	No	No	Yes

Table 6.10: S1 Efficiency Runs

Run ID	BaseEff (%)	Best Generated	Best Optimized	AveImpr (%)	TradeOff (ozft)(%)
S1AE	64.1%	73.6%	74.8%	6.1%	-3 (-8.9%)
S1BE	64.1%	72.4%	75.3%	2.8%	-2 (-5.6%)
S1CE	64.1%	71.6%	73.5%	2.6%	-3 (-8.9%)

Table 6.11: S1 BreakdownTorque Runs

Run ID	BaseTrq (ozft)	Best Generated	Best Optimized	AveImpr (ozft)(%)	TradeOff (%)
S1AT	39	49	50	3 (7.7%)	-4.3%
S1BT	39	47	51	2 (5.1%)	0.4%
S1CT	39	50	52	2 (5.1%)	-0.1%

Table 6.12: S2 Efficiency Runs

Run ID	BaseEff (%)	Best Generated	Best Optimized	AveImpr (%)	TradeOff (ozft)(%)
S2AE	78.0%	78.5%	78.5%	1.2%	-1 (-0.3%)
S2BE	78.0%	80.4%	80.6%	0.8%	1 (0.2%)
S2CE	78.0%	80.4%	80.8%	0.9%	0 (0.2%)

Table 6.13: S2 BreakdownTorque Runs

Run ID	BaseTrq (ozft)	Best Generated	Best Optimized	AveImpr (ozft)(%)	TradeOff (%)
S2AT	323	364	364	7 (2.2%)	0.2%
S2BT	323	356	382	24 (7.3%)	-0.8%
S2CT	323	360	384	28 (8.5%)	0.0%

In the Efficiency runs, we observe a larger number of turns resulting in a higher slot fill. Stator slot depths are larger where allowed to vary. Air gap in S1CE is also marginally higher. The rotor slot depth is smaller whereas the stator slot lip is always larger.

In the Torque runs, the auxiliary winding turns are higher but slot fills are lower on the whole. The stator slot depths are larger on the whole. The air gap is larger with the S1CT run.

6.1.3 Generators

Similar to the Polyphase and SinglePhase experiments presented above, we conducted an experiment with the Generator solver. However, in this case, we only present one complete experiment (i.e. only one experiment was considered as opposed to two each presented for the other solvers). The attributes of interest, here are Efficiency and Field Current, and the runs are labeled as G1AE, G1AF, and so on. The machine considered is shown in

Table 6.14: The Generator Experiment Considered

Specifications	**Generator (G1)**
Rated PP	40KW
No of Poles	4
Frequency	60 Hz
Voltage	480 V
Frame Size	285
Rotor/Stator Wire Type	Round
Attribute Targets:	
Efficiency (%)	86-100 %
Stator SlotFill (%)	$< 75\%$
Field Current at PP	$< 40 Amps$
Rotor Slot Fill(%)	$< 85\%$
Xd	< 5
Xd'	< 0.4
Xd"	< 0.3
Field Temperature	< 115
Stator Temp	< 115
Armature Current Density	< 6500
Field Current Density	< 5000

Table 6.14, and the three examples, A, B, and C, are shown in Table 6.15.

The G1 Efficiency Runs and G1 Field Current Runs are shown in Table 6.16, and Table 6.17 respectively.

6.2 Variability of Prediction

6.2.1 Importance of Variability in Presented Results

In Section 6.1 we presented results based on values reported by the respective solvers. We made an implicit assumption that the values reported were correct and reliable, and there was no variability associated with the predictions. This approach was consistent with the aims of the NDA, where we strive to design better electric machines using computer based design tools. A solver to compute the performance value accurately for any given set of design variables, is a basic requirement of the NDA approach. The circuit model solvers used in the NDA fit this requirement perfectly. However, upon careful reflection, it quickly

Table 6.15: Three Examples for Generators

Variables to Design	Example A	Example B	Example C
Core Length, #turns/pole #turns/coil, rotor wire GA stator wire GA	Yes	Yes	Yes
Stator Slots	No	Yes	Yes
Rotor Geometry, Stator ID & Rotor OD	No	No	Yes

Table 6.16: G1 Efficiency Runs

Run ID	BaseEff (%)	Best Generated	Best Optimized	AveImpr (%)	TradeOff (Amps)(%)
G1AE	89.0%	91.1%	91.9%	0.8%	0.5 (1.7%)
G1BE	89.0%	91.8%	92.4%	1.2%	0.8 (2.7%)
G1CE	89.0%	91.4%	92.7%	1.8%	5.3 (17.8%)

Table 6.17: G1 Field Current Runs

Run ID	BaseAmps (Amps)	Best Generated	Best Optimized	AveImpr (Amps)(%)	TradeOff (%)
G1AF	29.7	22.8	22.0	3.9 (13.1%)	0.6%
G1BF	29.7	23.5	21.7	4.2 (14.1%)	0.6%
G1CF	29.7	21.7	19.6	6.6 (22.2%)	1.1%

comes to light that our results are only as good or as accurate as the solver used to compute those results. If the solver reports incorrect results, or has large prediction errors associated with it, then the results from the NDA would also reflect similar errors.

What does that mean for the methodology developed in this thesis? It means that the methodology developed for the NDA is still applicable and acceptable as a viable means of designing electric machines! Circuit model solvers for analyzing electric machines are well developed and have been well honed over the past few decades. Moreover, the *same* solvers are used routinely in industry to design machines. Upon talking to experienced designers in MagneTek, we gathered that their solvers are very accurate and are considered very reliable for industrial design purposes. In most situations, the solver predicted values are attained by manufactured motors, if the designs are manufactured maintaining very tight tolerances. Most of the variability in production is associated with manufacturing tolerances.

Nevertheless, the performance of actual manufactured designs does vary from the solver predicted values (albeit marginally). Ideally, a solver should also report a confidence interval or some measure of error bounds with its predicted results. Many design changes are expensive to implement. Designers and managers should be able to account for prediction errors when suggesting design changes based on their solvers, or tools like the NDA. Since the solvers used in the NDA do not report any error bounds for predicted values, we will attempt to demonstrate our ideas (for improved design decision making using the NDA) with a very basic and simplified approach for computing prediction errors.

6.2.2 Sources of Variability

There are several sources of variation in an electric machine design. We broadly classify them in three categories: variation due to material properties, variation due to dimensional tolerances, and variation resulting from other mechanical sources. Variation in material properties is dependent on the material and the manufacturing processes of those materials. Examples include conductivity of copper/aluminum, and permeance of steel. While most of these properties are fairly stable, there may be some variation based on the manufacturing processes used to produce the copper, aluminum, or steel for use in electric machines. Some

material properties also vary with temperature. MagneTek's solvers do account for some of this temperature variation.

Variation due to manufacturing tolerances is perhaps the only kind of variation which is possible to control to some extent (of course, maintaining extremely tight tolerances can be expensive). This will be the kind of variation which we will attempt to quantify and base our prediction errors on.

Other mechanical sources of variation include friction, bearings, and lubrication. Again, this is hard to predict in an electrical solver. Predicting material property variation and mechanical element based variation would require separate analytical models. Even if such models were available, it would be hard to predict these quantities accurately.

Hence, the only kind of variation which lends itself to (relatively) easy estimation is the variation based on dimensional tolerances. We will develop a simple model to predict variability based on these errors, and use it to validate some of our results presented earlier.

6.2.3 2^k Factorial Experimental Design

One of the problems of studying the effects of certain variables on a designated response, is the presence of interactions among variables. When strong interactions are present among variables, then the effect of one variable on the response is dependent on the values or levels of other variables present. General factorial experiments attempt to study all possible combinations of all explanatory variables, and their effects on the response.

A simplification of general factorial experiments is the 2^k factorial experimental design. A good treatment of general factorial design and 2^k factorial designs may be found in standard statistics texts like [50]. We will only outline the approach here.

In 2^k factorial experiments, every variable varies over exactly two different values or levels. If there are k such variables, this produces exactly 2^k combinations (and hence the name). Since the number of combinations can be very large for a large number of variables, we will attempt to design a complete 2^k factorial experiment with a restricted set of variables. So, for our case, $k < n$, the total number of variables.

To understand our simple scheme, let us consider an example with two variables (2^2

factorial experiment). Let us say that rotor outer diameter and stator inner diameter are the only two variables of interest. Tight manufacturing tolerances are maintained on both of these variables. However, they may still vary by one or two thousandths of an inch. So let us say the nominal value of rotor OD and stator ID are $3.599in$ and $3.626in$ respectively. Now, if each of these may vary by $\pm 0.001in$ due to manufacturing, then we have estimates on the lower and upper levels of the two variables of interest. In this case, a 2^2 factorial experiment will use two levels of rotor OD as $3.598in$ and $3.6in$, and the two levels of stator ID as $3.625in$ and $3.627in$. Examining the responses for each of these four combinations gives us an idea of the variation in our predicted responses based on the variation in two explanatory variables.

The 2^2 factorial experiment may be schematically shown as in Figure 6-1.

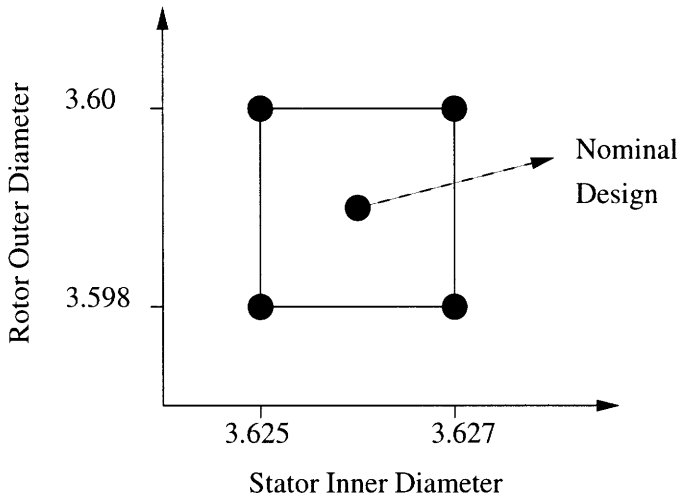


Figure 6-1: A 2^2 Factorial experiment

The four combinations form the corners of a square around the nominal design point. When the same concept is extended to 2^k factorial experiments, the combinations form a k dimensional hypercube around the nominal design point.

6.2.4 P1 Example

As mentioned above, the number of combinations can get really large for large k in a 2^k factorial experiment. Since our aim is to demonstrate this methodology as a possible means of estimating prediction errors for a solver, we will restrict ourselves to a smaller subset of the total number of variables for both the examples presented in this section.

We would like to reiterate that in industrial design practice, more detailed and sophisticated error limit computations may be in order. Ideally, these computations would be performed by the solver for every analysis.

For the purposes of demonstrating this methodology, we choose two of the examples presented in Section 6.1. In this subsection, we will use the P1 series of experiments, and in the next subsection, we will use similar runs for the P2 series of experiments.

For the P1 example, we restrict ourselves to 11 variables, thereby yielding a 2^{11} factorial experiment. This represents 2048 combinations. 11 was chosen rather arbitrarily – more than 11 variables leads to a really large number of combinations. Besides, 11 variables are enough to capture the most important variables pertaining to the rotor and stator laminations. These represent the variables subject to maximum manufacturing variations. Variables pertaining to windings, for example, were not chosen since it is unusual to make an error with discrete variables ("manufacturing variation" would never result in 37 coils instead of 36!). Material properties, and other mechanical elements are not considered in our experiments, as explained earlier.

For the P1 series of experiments, the eleven variables chosen are: stator slot depth, stator slot top width, stator slot opening, stator slot bottom width, top rotor slot depth, top rotor slot top width, bottom rotor slot top width, bottom rotor slot depth, middle rotor slot depth, stator ID, and rotor OD. The manufacturing tolerances on each of these variables are $\pm 0.001 in$. The exact tolerances on each of these variables cannot be used here for proprietary reasons. However, $\pm 0.001 in$ presents a good estimate.

The lower and upper levels on each of these variables are coded as "0" and "1" respectively. A part of the 2^{11} factorial experiment is shown in Table 6.18. The 11 variables are designated as x_1, x_2, ..., x_{11}. Efficiency and Breakdown Torque values are reported for

Table 6.18: 2^{11} Factorial Experiment for the P1 motor

No	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	Effn (%)	Trq (lbft)
1	0	0	0	0	0	0	0	0	0	0	0	93.77	1353
2	0	0	0	0	0	0	0	0	0	0	1	93.76	1360
3	0	0	0	0	0	0	0	0	0	1	0	93.78	1346
4	0	0	0	0	0	0	0	0	0	1	1	93.77	1352
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2047	1	1	1	1	1	1	1	1	1	1	0	93.77	1353
2048	1	1	1	1	1	1	1	1	1	1	1	93.77	1360
											MIN:	93.75	1344
											MAX:	93.79	1369

every combination. In the end, minimum and maximum values are computed for efficiency and breakdown torque over all the 2048 combinations.

We observe that to the first decimal place, efficiency does not vary from the nominal value for this set of experiments (within 0.05% of the base design). Breakdown torque varies from 1344 lbft to 1369 lbft which represents a variation of $\pm 0.88\%$ about the nominal value of 1356 lbft. Using these numbers as error limits, our P1 results presented in Section 6.1 are definitely significant. With P1AE, we predicted a 0.2% increase in efficiency, and in P1AT, a 353 lbft (26%) increase in torque.

6.2.5 P2 Example

As a second example to demonstrate our methodology, we choose the P2 series of experiments from Section 6.1. Again, like the P1 example presented above, we choose a 2^{11} factorial experiment. Again the variables are labeled as x_1, x_2, ..., x_{11}, and the two levels of all the variables are coded as "0" and "1". A part of this 2^{11} factorial experiment is shown in Table 6.19.

We observe that to the first decimal place, efficiency varies from 85.6% to 86.1%, or from -0.3% to $+0.2\%$ of the nominal value. Breakdown Torque varies from 42 lbft to 44 lbft or $\pm 2.3\%$ from the nominal value. Using these numbers as error limits, the P2 results presented in Section 6.1 are significant. In P2AE, we predicted a 1.9% increase in efficiency,

Table 6.19: 2^{11} Factorial Experiment for the P2 motor

No	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	Effn (%)	Trq (lbft)
1	0	0	0	0	0	0	0	0	0	0	0	85.9	43
2	0	0	0	0	0	0	0	0	0	0	1	86.0	43
3	0	0	0	0	0	0	0	0	0	1	0	85.8	43
4	0	0	0	0	0	0	0	0	0	1	1	85.9	43
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2047	1	1	1	1	1	1	1	1	1	1	0	85.6	43
2048	1	1	1	1	1	1	1	1	1	1	1	85.7	43
											MIN:	85.6	42
											MAX:	86.1	44

and in P2AT an 8 lbft (18.6%) increase in torque.

6.3 Sensitivity Analysis

In Section 6.2, we examined the significance of presented results assuming that manufacturing variation was limited to $\pm 0.001in$ on all lamination geometry. In this section, we will perform a different kind of sensitivity analysis, and look at the change in target attribute values with respect to changes in lamination geometry parameters. We would also try to obtain an understanding of the variables which might be considered critical (from manufacturing tolerance standpoint) in terms of affecting target attribute values significantly.

In other words, Section 6.2 dealt with the absolute variation in a target attribute (e.g. efficiency) due to manufacturing variation. Here, we will examine the variables which affect these changes in the target attribute value. This would also help us identify the variables, if any, whose manufacturing tolerance limits might be critical to maintaining significantly accurate values of the target attribute.

While sensitivity analyses may be performed for all designs, whether optimal or sub-optimal, we have chosen to focus on the optimum designs produced by the NDA. As in Section 6.2, we will only focus on polyphase examples.

Sensitivity analysis may be performed on any design by examining the vector $\partial f/\partial \mathbf{x}$ at the desired design point, where f denotes the objective function (in this case the primary

attribute), and $\mathbf{x}$ denotes the vector of design variables. Theoretically, for an unconstrained problem, $\partial f/\partial x_i = 0$ for all design variables (for $i = 1, \ldots, n$). However, this is not necessarily true in our case for two reasons:

- **Constraints**: Objective functions need not have all partial derivatives equal to 0, if they hit a constraint at the optimum point. Constraints can be of two types: Bound Constraints and Non-Linear inequality constraints (we do not have linear constraints or non-linear equality constraints in our problem). For a complex multidimensional problem like ours, it is hard to determine analytically if any constraint (especially non-linear) has been hit at any particular point. Perhaps the only way to check is if one of the constraints is at or very near its limit. However, this may not be as straightforward as it may appear. Presence of discrete variables makes it hard to detect if a constraint is near its limit. In such cases, it is possible, that a constraint has been hit, but the constraint value is not near its specified limit. For example, let us say we have one non-linear constraint whose value should always be below 3.6. Now, if due to the presence of discrete variables, this constraint can only assume values in increments of 1/2, then at the value of 3.5, effectively, the constraint has been hit but the value of 3.5 is not equal to the initial limit on the constraint value (3.6). In such cases, we always make a judgment if a constraint boundary has been hit. In any case, we definitely conclude that constraints can lead to non-zero partial derivatives in the vicinity cf the optimum.

- **Model Inaccuracies**: For our purposes in this Section, we will determine partial derivatives (or gradients) using the MARS models developed in Chapter 4. Like all other models, this model is not 100% accurate at all locations in the design space. Even though we have a very good model of the space, there are bound to be some inconsistencies between the actual space and the model. Such inaccuracies also sometimes lead to non-zero partial derivatives. For our purposes, however, we will use the MARS model to ascertain all gradients, and use it to report any non-zero derivatives.

In the remainder of this section, we will focus on the partial derivatives which are

non-zero. By definition, the objective function is not sensitive to the variables which have zero slopes (at the optimum point). There may be instances where a designer would be interested in the behavior of the objective function in a neighborhood in the vicinity of the optimum location (even for variables whose partial derivatives are zero at the optimum). This information is useful to determine the nature of changes resulting from manufacturing tolerances. This was tackled in Section 6.2 where we tried to vary all variable values in all possible combinations (within their expected tolerance limits). What remains is to examine all variables which have non-zero derivatives. Hence, in the tables below, we will focus only on variables whose partial derivatives are non-zero at the optimum point.

Table 6.20 shows variables with non-zero partial derivatives, for the P1 series of designs (from Section 6.1). The three optimum designs (optimizing efficiency), P1AE, P1BE, and P1CE are shown in columns. For each design, we tabulate the partial derivative in the $\partial f/\partial x_i$ column. We have only chosen variables whose partial derivative is non-zero. Other variables have not been shown. The Δx_i column gives us an indication of the magnitude of change required for that variable to cause a significant change in efficiency (target attribute). Here, as in Section 6.1, we have assumed that efficiency is significant to the first decimal place. Hence to cause a 0.1% change in efficiency, what is the magnitude of change or variation desired in any variable x_i? This desired change is shown in Δx_i. If the magnitude of the change is high (higher than the manufacturing tolerance on a variable), then the design is safe, since variation within tolerance limits is unlikely to affect the predicted efficiency. However, if the magnitude of the required change is low (within or close to tolerance limits), then it is quite likely that the predicted efficiency will vary by a significant amount (more than 0.1%).

Some of the entries in Table 6.20 are blank. This indicates that those variables were not allowed to vary for that particular example. For instance, in P1AE, Stator Slot Bottom Radius is not allowed to vary (refer Section 6.1, Table 6.2), and hence the derivative and Δx_i columns are blank for this variable.

We notice that none of the variables in Table 6.20 are critical, since Δx_i is always more than ± 0.001. In fact, for the P1 series of designs, none of the variables even come

Table 6.20: Sensitivity Analysis for optimized P1 motors

Variable Name	P1AE		P1BE		P1CE	
	$\partial f/\partial x_i$	Δx_i	$\partial f/\partial x_i$	Δx_i	$\partial f/\partial x_i$	Δx_i
# Wires (1)	0	∞	-0.24	-0.42	-0.65	-0.15
# Wires (2)	-0.39	-0.25	-0.66	-0.15	-0.57	-0.18
Wire Size (2)	0.25	0.4	-0.09	-1.01	0.20	0.49
Core Length	-1.01	-0.10	-0.56	-0.18	-0.90	-0.11
Stator ID	-	-	-	-	-9.66	-0.01
SS Depth	-	-	1.21	0.08	0.86	0.12
SS Top Width	-	-	1.47	0.07	7.29	0.01
Rotor OD	-	-	-	-	10.10	0.0099
Top RS Bot Wid	0	∞	0	∞	-2.79	-0.04
Top RS Depth	-1.86	-0.05	-3.63	-0.03	-2.00	-0.05
Bot RS Top Wid	-0.63	-0.16	-0.62	-0.16	-2.39	-0.04
Bot RS Bot Wid	0.17	0.58	0.06	1.65	-1.87	-0.05
Bot RS Depth	-0.25	-0.4	-0.85	-0.12	-0.39	-0.26
End Ring 1 A	-	-	-0.44	-0.23	-0.22	-0.45
End Ring 1 D	-	-	-0.08	-1.23	-0.25	-0.39

close to being considered critical! This observation is also corroborated by the fact that in Section 6.2, none of the possible changes in variable values (within tolerance limits) produced a significant change in efficiency.

To double check some of the results presented in Table 6.20, we manually varied the parameters Δx_i to see if this predicted value did produce the expected 0.1% change in efficiency. We randomly picked 3 variables, and varied them one at a time from the P1CE optimum design. These variables were Top RS Depth (varied by $-0.05in$), Bot RS Bot Width (varied by $-0.05in$), and Bot RS Top Width (varied by $-0.04in$). The P1CE optimum design had an efficiency of 94.2%. Varying these three variables each produced an efficiency of 94.12%, 94.14%, and 94.12%, i.e. 94.1% (to the first decimal place).

Table 6.21 is a similar table for the P2 series of motors. The P2AE, P2BE, and P2CE series of optimum designs are shown here. As in Table 6.20, variables which are not allowed to vary are blank, and none of the variables are critical (the minimum variation required for 0.1% change in efficiency is more than manufacturing tolerance of $\pm 0.001in$).

However, one notices an important difference here. Although none of the variables are

Table 6.21: Sensitivity Analysis for optimized P2 motors

Variable	P2AE		P2BE		P2CE	
Name	$\partial f/\partial x_i$	Δx_i	$\partial f/\partial x_i$	Δx_i	$\partial f/\partial x_i$	Δx_i
Wire Size (1)	1.38	0.07	1.11	0.09	1.17	0.09
Core Len	-3.88	-0.03	-3.57	-0.03	-3.48	-0.03
C1	-0.43	-0.23	-0.42	-0.24	-0.39	-0.25
Stator ID	-	-	-	-	-6.24	-0.02
SS Depth	6.89	0.01	11.32	0.009	7.85	0.01
SS Top Width	37.65	**0.002**	4.52	0.02	10.56	0.009
SS Lip	-12.00	-0.008	-0.73	-0.14	-2.75	-0.04
SS Bot Rad	55.43	**0.0018**	48.71	**0.002**	40.92	**0.002**
Rotor OD	-	-	-	-	2.72	0.04
RS Depth	5.75	0.02	5.91	0.02	6.72	0.01
RS Top Wid	0.53	0.19	-34.28	**-0.002**	6.52	0.01
RS Lip	24.94	0.004	21.38	0.005	-14.13	-0.007
RS Bot Rad	46.33	**0.0022**	45.03	**0.0022**	74.18	**0.0014**
End Ring 1 A	-	-	-1.37	-0.07	-2.96	-0.03
End Ring 1 D	-	-	-0.35	-0.29	-0.22	-0.46

close to $\pm 0.001in$, a small number of them do come close e.g. the bottom radii on the stator and rotor slots, are always in the vicinity of $\pm 0.002in$. Hence, with this analysis, the designer can get a sense of which variables are important from the manufacturing tolerance standpoint. Some attention has to be paid to these variables, and their tolerance limits. Excess variation in these variables, beyond their respective tolerance limits, may be cause for concern. Some of such variables, and their values are shown in bold face type in Table 6.21.

Similar to the P1 case, we picked 3 variables to vary from the P2CE optimum design to double check if the predicted Δx_i did produce a change of 0.1% in efficiency. The variables varied were SS Top Width (varied by $0.009in$), SS Bot Rad (varied by $0.002in$), and RS Lip (varied by $-0.007in$). Again, it was verified that all three changes produced efficiencies which were within 0.1% of the efficiency of the P2CE optimum design.

We must emphasize, however, that in the foregoing analysis, we have only considered one variable at a time. We have only considered the impact of one variable x_i, and the change required, Δx_i for it to cause a significant change in efficiency. We have not considered how variations in combinations of different variables might impact the efficiency.

Some of this information may be obtained by the designer from the analysis in Section 6.2 where we allow every combination of variables. Significant changes in efficiency for one particular combination, might point a designer in the appropriate direction for ascertaining the sensitivities.

Before we conclude this section, we must mention that the feature presented in this section is not implemented together with the user interface. The output from sensitivity analysis is sent to a flat ASCII file which presents all the information in a tabular fashion.

6.4 Cost Implications

In Section 6.1 we presented our results and in Section 6.2, we talked about the accuracy of those results. If the NDA is used in an industrial environment, the next natural question is: Should we go ahead and implement the designs suggested by the NDA? How expensive is it to implement those design changes? Would it offer any benefits for the customers? When should one decide to change a particular line of motors? These would be almost inevitable questions faced by engineers and managers in a firm.

In this Section, we will think through some of these issues. Ideally, the solvers used should also let us estimate manufacturing costs for a motor being analyzed. This would make it simpler for a designer to use the NDA (or the solvers alone) effectively. However, in our case, the solvers do not output these numbers. In real life, cost estimation is performed using other tools and estimation techniques. Again, for proprietary reasons, we do not have access to all this information. Nevertheless, we will try to use some assumptions and educated guesses to arrive at a rough estimate of cost for a motor. This would help us follow through with the rest of our analysis, and demonstrate one possible way of thinking about the issues at hand.

6.4.1 Motor Cost Estimation

Motor cost estimation is a complex task. [111] tackles these issues for one particular factory. While the ideas are fairly generalizable, details would differ greatly from one factory to

another. [72] was able to use the cost simulator of [111], and use cost as an attribute.

As mentioned above, this is the ideal case, when the solver can estimate the cost of manufacturing any motor which is being analyzed. In the absence of such cost estimators, and proprietary information for estimating costs, we will attempt to develop a very simple scheme to estimate the cost of a motor. We will acknowledge upfront that this scheme is not meant to be accurate or sophisticated. It serves merely as a stepping stone for demonstrating how we can think about cost related decisions.

We know that the major contributors to material cost in a motor are the copper (wires) and steel (core) in a motor. Of course, other determinants of cost are the aluminum used, frame, bearings, and manufacturing or assembly related costs. We will make some simple assumptions here. Let us say that the cost of iron and copper together, are a fixed percentage of the total manufacturing cost, and the cost of manufacturing is another fixed percentage of the list price of a motor. In this fashion, the cost of iron and copper can be related linearly to the list price (or selling price) of a motor. We know that iron costs roughly \$0.3/lb and copper costs about \$2.0/lb. This leads us to a direct linear relationship between the list price and the mass of copper and iron used in a particular motor.

$$Price = a_0 + a_1 * m_{fe} + a_2 * m_{cu} \tag{6.1}$$

where m_{fe} is the mass of iron, and m_{cu} is the mass of copper in any particular motor. The per unit mass cost of iron and copper are embedded in coefficients a_1 and a_2 respectively.

We will use linear regression to estimate the coefficients of this relationship. Let us demonstrate this using the P1 and P2 motors from Section 6.1.

P1 Motor

This is a 150Hp, 4 pole, 460V, 444 Frame motor. We look at MagneTek's catalog of available motors and find all motors which are 150Hp, 4pole, 460V, and 444 or 445 Frame. 445 Frame motors are somewhat larger but are sufficiently similar to give us more data points for our regression (they are also proportionately more expensive). We find 17 such motors in the Spring 1997 catalog. The catalog also lists their weight, and list price.

Table 6.22: Regression Data for Motors Similar to P1

Price ($)	Weight (lb)	m_{fe} (lb)	m_{cu} (lb)
5968	1029	623.68	62.37
5968	1025	621.25	62.13
6863	1250	757.63	75.76
6863	1250	757.63	75.76
7362	1318	798.84	79.88
6863	1250	757.63	75.76
6863	1250	757.63	75.76
7362	1318	798.84	79.88
10144	1644	996.43	99.64
12649	1767	1070.98	107.10
11158	1750	1060.68	106.07
12274	1773	1074.62	107.46
13374	1992	1207.35	120.74
12274	1773	1074.62	107.46
13374	1992	1207.35	120.74
13729	1930	1169.77	116.98

At this stage, we do not know what proportion of the listed weight comes from iron or copper. Let us make another assumption here. Let us say 2/3 of the listed weight comes from iron and copper. Moreover, the iron and copper masses are in a fixed ratio on an average. We let the NDA generate 200 designs which are 150Hp, 4pole, 460V and 444Frame. From these 200, we estimate the average ratio of copper mass to iron mass for such motors. From this exercise, we estimate the average ratio of copper mass to iron mass to be 0.1.

These simplifications let us arrive at the iron and copper mass in a motor given the total weight. This information for all the 17 motors found in the catalog is shown in Table 6.22. The price and the weight columns are obtained from the catalog, and the Iron and Copper Mass columns are obtained using the above simplification from the total weight.

Performing a linear regression with the data in Table 6.22, we estimate the coefficients of Equation 6.1:

$$Price = -3804.8 + 8.2 * m_{fe} + 63.19 * m_{cu} \tag{6.2}$$

Table 6.23: Regression Data for Motors Similar to P2

Price ($)	Weight (lb)	m_{fe} (lb)	m_{cu} (lb)
312	67	38.84	5.83
390	87	50.43	7.57
330	64	37.10	5.57
306	81	46.96	7.04
367	81	46.96	7.04
459	92	53.33	8
569	126	73.04	10.96
409	95	55.07	8.26
409	95	55.07	8.26
312	67	38.84	5.83
390	87	50.43	7.57
367	81	46.96	7.04
459	92	53.33	8
671	107	62.03	9.30
622	107	62.03	9.30
389	68	39.42	5.91

The regression itself is fairly accurate with an R-squared of 0.97, and t-statistics of -5.68, 65535, and 65535 respectively for each of the three coefficients. We would have accepted this regression model even if the R-squared value was 0.7, and the t-statistics had been as low as 2.

P2 Motor

In order to present another example with our proposed methodologies in this section, we perform the exact same regression with the P2 motor. This time, we identify 16 motors from the catalog which are 3Hp, 4 pole, 460V. We accept 56 and 182 Frame motors. Again, 182 frame is considerably larger and proportionately more expensive but it is similar enough for us to use for obtaining additional data points in our regression model. The ratio of copper mass to iron mass this time is 0.15. We present the regression data in Table 6.23 which is exactly analogous to Table 6.22.

Performing a linear regression to fit the coefficients of Equation 6.1, we obtain:

$$Price = -52.5 + 0 * m_{fe} + 62.57 * m_{cu} \tag{6.3}$$

Again, the regression model itself is fairly significant, with an R-squared of 0.7, and t-statistics of -0.6, 65535, and 65535. The constant coefficient seems to have a high standard deviation, but otherwise, this regression model appears to be acceptable.

Using our regression models of Equation 6.2 and 6.3, we can estimate the list prices of motors similar to P1 and P2 respectively.

6.4.2 Efficiency Improvement – Customer Viewpoint

Having developed a simple framework for estimating the cost of a motor, let us address some decision making issues. In Section 6.1, the NDA suggested improvements for both P1 and P2 motors. Do those improvements make sense? In this subsection, we will deal with the customer viewpoint, and examine whether these changes would imply any significant benefits for the customer. In the following subsection, we will examine whether it is cost effective for the firm to make those changes.

Again, here we will illustrate our ideas with the P1 and P2 examples. Analyses for other results follow similarly.

Let us say we compute the price of the base design, and all the final optimized designs suggested by the P1E and P2E series of experiments. Here, we will use Equation 6.2 and 6.3 for this purpose. For the optimized design from the P1AE run, the iron weight is 643 lbs and the copper weight is 80.36 lbs. From Equation 6.2, we obtain the cost of this design to be \$6547.22. Following the same procedure, the cost of the base design is \$5556.98. This means the P1AE optimized design is \$990.24 more expensive. Is that extra cost justifiable for the customer?

It might be. This design does have a higher efficiency and hence, the running cost of the motor would be lower. If this motor is run 24 hours a day, and if electricity costs \$0.1 per kilowatthour, then the daily running cost of the base design is \$286.31, and the running

cost of the P1AE optimized design is $285.70. If this motor is run for 300 days in year, then the annual savings add up to $182.36! This is definitely substantial.

However, the customer would have to pay $990.24 extra for the optimized motor up front. How long does it take for the extra cost and savings to balance out? In other words, what is the payback period? We again make a few assumptions here. Let us say the interest rate is 8% per year (we just want the borrowing rate to be higher than the risk free or treasury bill rate which is around 5-6%) and the interest is compounded annually. Of course, it is simple to modify these assumptions and plug in "real" numbers for a company. Next, we use the annuity formula to compute the payback period. The annuity formula may be found in all Corporate Finance texts like [13]. (Alternatively, the interested reader can derive this formula starting from expressions for the sum of a Geometric Progression Series.)

$$Cost = \frac{C}{r}(1 - \frac{1}{(1+r)^t}) \tag{6.4}$$

where $Cost$ is our extra cost incurred upfront, C is our annual saving from using the optimized motor, r is the interest rate (we assumed 8% per annum), and t is the number of years (we are compounding interest annually). Rearranging this equation, we obtain an expression for the payback period t:

$$t = \frac{log(\frac{C}{C - Cost*r})}{log(1+r)} \tag{6.5}$$

For the P1AE optimized motor, the payback period is 7.4 years, which may be on the high side. For the P1CE optimized design, the payback period is 3.2 years. This may not be too high considering that the life of these motors is at least 15-20 years.

Similarly, we compute the costs and payback periods for all the optimized designs from the P1E and P2E series examples. These results are tabulated in Table 6.24. The Price column is the calculated list price from Eq 6.2 or 6.3, and the Cost Diff column shows the difference between the list prices of the optimized motor and the base motor.

Hence, we observe that for both P1 and P2 motors, it makes sense for the customer to

Table 6.24: Costs and Payback Periods for P1E and P2E series designs

Design ID	m_{fe} (lbs)	m_{cu} (lbs)	Price ($)	Cost Diff($)	Effn (%)	Exp/day ($)	Save/yr ($)	Payback Period (yrs)
P1Base	643	64.69	5556.98		93.8	286.31		
P1AE	643	80.36	6547.22	990.24	94	285.70	182.36	7.4
P1BE	643	81.12	6595.25	1038.27	94.2	285.10	364.34	3.4
P1CE	643	80.42	6551.01	994.03	94.2	285.10	364.34	3.2
P2Base	45.3	5.38	284.13		85.9	6.25		
P2AE	45.3	6.52	355.46	71.33	87.7	6.12	38.54	2.1
P2BE	45.3	7.2	398.00	113.88	88	6.10	44.81	2.95
P2CE	45.3	6.35	344.82	60.69	88	6.10	44.81	1.5

pay for the optimized motor. Before we end this subsection, we would like to re-iterate that these results may vary when more sophisticated costing schemes are used in place of our simple cost models.

6.4.3 Efficiency Improvement – Manufacturing Viewpoint

We have demonstrated that the improvements suggested in Section 6.1 are definitely significant for the customer. The question arises: is it cost effective for the company to manufacture the desired motor? The simple answer is that it is cost effective for the company if the demand is high enough.

Let us pick the optimized motor from P1AE again. From Equation 6.2, the price estimate for this motor is $6547.22. Assuming that this figure is accurate, it sells for $990.24 more than the current design, which sells for $5556.98.

But then, what is the gross profit for the company on these motors? Again, for the lack of proprietary information, let us estimate this number by looking at MagneTek's annual report. For FY 1997, MagneTek's total sales totalled $1.19 Billion and its gross profit was $239.9 Million which is about 20% of sales. Let us use this number for our rough estimates. Let us also assume that the list price for a MagneTek motor is the suggested retail price and includes dealer or vendor markup (say 25%). This markup would be eliminated when

the motor is sold directly by MagneTek to Original Equipment Manufacturers. Using these numbers, we estimate the ex-factory sale price of the base P1 motor to be $4445.58, and of the P1AE optimized motor to be $5237.78. The gross profits on these motors are then 20% of these prices i.e. $889.12 and $1047.56. Hence the gross profit differential is $158.44 per motor.

This definitely looks encouraging, since the company would make an additional gross profit of $158.44 per motor. But what capital investments might be needed for producing this motor? The P1AE design suggested changes to the rotor slot geometry. Again, for the lack of proprietary knowledge, let us assume that the tooling needed for creating different slot geometries costs around $100,000. This cost estimate is for the tooling needed for creating slot geometries. If the lamination diameters are to be changed (e.g. in the P1CE examples), the capital investment required is of the order of $500,000. For the P1AE example, the extra investment required is about $100,000 and the gross profit increase is $158.44 per motor. Hence, an order size of 631 might be required to justify this capital expenditure. While such a demand may or may not come from a single customer (Original Equipment Manufacturer), if the estimated demand is high enough, then this investment is well justified.

In Table 6.25, we list our summary calculations for the P1 and P2 series of experiments. List Price is the list price calculated using Eq 6.2 and 6.3. Sale Price is the price without the vendor markup. Gross Profit is the gross profit from selling that motor (20% of sale price). Extra G.P. is the difference between the gross profit for that motor and the gross profit for the existing base design. Investment is the capital investment needed to implement the suggested design changes, and Volume is the target order size to break even with this additional investment.

A note of caution! In Table 6.25 while computing volume or order size requirement for break-even analysis, we have assumed that the gross profit ratio remains constant for all the machines. This need not necessarily be so. Since different designs offer different benefits for the customer, their prices can be adjusted/increased accordingly. This, in turn, would lower the order size (Volume) requirement to break-even. For example, the P1CE design

Table 6.25: Manufacturing Costs and Order Sizes for P1E & P2E Experiments

Design ID	List Price ($)	Sale Price ($)	Gross Profit ($)	Extra G.P. ($)	Investment ($)	Volume
P1Base	5556.98	4445.58	889.12			
P1AE	6547.22	5237.78	1047.56	158.44	100,000	631
P1BE	6595.25	5276.2	1055.24	166.12	200,000	1204
P1CE	6551.01	5240.81	1048.16	159.04	500,000	3144
P2Base	284.13	227.30	45.46			
P2AE	355.46	284.37	56.87	11.41	200,000	17523
P2BE	398.00	318.4	63.68	18.22	200,000	10977
P2CE	344.82	275.86	48.56	3.1	400,000	129199

offers a relatively low payback period for the customer (refer Table 6.24). If we increase the payback period for the (additional cost of the) P1CE motor to 7 years, then the list price of the motor could be around $7500. In that case, the sale price is $6000, the gross profit is $1807.35, which is $918.32 over the gross profit from the base design. In that case, with an investment of $500,000, the order (demand) size is only 545 to break even! Such pricing changes are very common in a real life company.

We would like to reiterate that some of these presented results and values are likely to vary as actual numbers are used instead of the estimated/assumed quantities.

6.4.4 Breakdown Torque Improvement

It was relatively straightforward for us to think of efficiency in terms of concrete dollar amounts. It is harder for us to think of breakdown torque in such simple terms. Breakdown torque (the maximum torque produced by a machine) is a measure of strength of the machine. A customer would place a lot of value on this attribute if the load on the motor is expected to vary substantially. Moreover, breakdown torque is proportional to the square of the voltage ([35], [2]). In applications where the voltage might vary significantly, the breakdown torque becomes an important attribute.

Still, it is hard to quantify the benefits of breakdown torque. The value of having

additional torque varies by application, and hence is different for different customers. One way to quantify breakdown torque is in terms of expected "down time" for the application. For example, if the P1 motor (150Hp) is used in a luggage conveyer at an airport, down time may not be a very critical issue. It is possible to switch to other conveyers. However, if the motor is used in a conveyer out of a blast furnace in a steel plant, the down time may be an extremely critical factor. Not only is it expensive to stop the conveyer from such a high production plant, it is also critical not to disrupt the blast furnace operation for a long time. It takes an extremely long time to start a blast furnace once it has been shut down, and down times of such magnitude are extremely costly!

Hence, depending on the application, the customer would be willing to pay extra for breakdown torque, or sacrifice other attributes in favor of additional breakdown torque.

6.5 Improved Decision Making

Two of the original aims of the NDA, were to improve the development lead time for new designs, and to afford multiple scenario analyses thereby improving the quality of the decision process. We have successfully demonstrated that the NDA meets these goals. In addition, the NDA can be useful as an aid in other decision making scenarios. We have shown in Section 6.1 and Section 6.4 that information from the NDA can be used in conjunction with other company specific information to outline the cost benefits for a customer and to decide pricing strategies and production decisions for a company. A quick reflection indicates that these ideas could be extended to include decision making in other arenas. Some of the areas where the NDA affords improved decision making for the company and its management are outlined below:

- Cost benefits to customers. This was demonstrated above in Section 6.4.
- Cost implications for changing manufacturing dies and other tooling. This was also tackled above in Section 6.4.
- Cost effective alternatives to customers. Since the NDA offers 200 design alternatives as a result of the Design Sub-Space Identification step (Chapter 3), some of these 200

designs could potentially serve as low cost substitutes for the customer's requirements with marginal compromise in some attributes. Some such potentially low cost designs were observed in some of the runs with the NDA.

- Fast appraisal of potential benefits of changing a manufacturing line. Typically, in a company, every line of motors is revamped every 4-5 years (other motors are changed in the interim). So this upgrading is a continual process for a company. When a new line of motors is planned, designers typically spend about 1-2 months trying to design the optimum motor for the new line. With the NDA, this development lead time can be shortened. But more importantly, since the time for analysis is reduced considerably, the NDA can be used up-front for all existing lines to determine which would be the most profitable line of motors to change. Such a decision aid would be of immense value since the demand patterns change constantly and it is critical to make the best investment (upgrading) decision based on current market conditions.

- Strategic Planning. The same idea can be easily extended for strategic (long term) planning. The entire existing line of motors could be analyzed with the NDA, and strategies could be developed for different market conditions in the future. For example, if one market situation demands one kind of investment decision (change of motor manufacturing line), and another market condition demands a different change in another manufacturing line, then strategy planners could use the NDA to analyze the scenarios well in advance, and take a quick and informed decision, when market conditions actually change.

- Competitive Advantage. If the customer response time is low, it could serve as a potential competitive edge against competitors. The NDA could be used to obtain very quick estimates for meeting customer requirements, and cost and investment issues related to the desired change.

6.6 Recapitulation

In this chapter, we presented results and analyses from using the NDA, which was developed based on the description in earlier chapters. Section 6.1 shows a number of examples using different solvers. Comprehensive experiments are devised which illustrate the methodology and provide some insights into the tradeoffs associated with multiple objective optimization. Section 6.2 attempts to tackle the important issue of variability in predicted quantities. Section 6.3 presents a sensitivity analysis on some of the results presented in Section 6.1. Section 6.4 deals with the cost implications of design changes. We demonstrated how results from the NDA could be used in conjunction with other company specific information, to obtain very practical and meaningful information for the company. This includes cost benefits to the customer and cost and investment implications for the company's manufacturing unit. We conclude this chapter with a list of potential benefits from using the NDA in the manner shown. In the next chapter, we conclude this thesis with a brief summary, our conclusions, and suggestions for future work.

Chapter 7

Summary, Conclusions, and Suggestions for Future Work

7.1 Thesis Summary

Induction Motors and Generators are complex machines which are difficult to design. Designers usually design them using their intuition and experience. Consequently, this design process takes a long time, sometimes to the order of a few weeks. If this design process were automated, it would afford considerable time savings, and could potentially result in better solutions. Hence, a need was felt in the industry for such an automation tool.

This project was started with the aim of developing a design tool which would offer design automation and optimization capabilities. This tool would also allow a multiple scenario analysis, improve the development lead time, and help react fast to customer requests. The design tool would be implemented to develop an industrial strength software system.

Even though this thesis concentrates on induction motors and generators, the methodology presented here should be applicable to a large number of engineering artifacts.

The problem of generating a design solution based on the user's requirements is a non-trivial problem. It occurs very frequently, in different forms, in many disciplines. A number of researchers have worked on this general problem concept and have developed techniques

for solving such problems. There is no universal solution or recipe for success, however. Most solutions are either problem specific or have certain strengths and weaknesses which make them suitable for only certain classes of problems. Our problem is very large (large number of variables), has multiple objectives, is highly non-linear, has a number of coupled equations, and has a number of categorical and discrete numerical variables. Most of the techniques found in literature do not attempt to tackle these issues. A two-step methodology has been developed in this thesis to solve our particular problem.

Chapter 3 presents the first step of the methodology. Monte Carlo design synthesis is used to identify a region in design space which is likely to contain designs matching the user's requirements. A Gaussian distribution of variables is used to identify this region. The design sub-space identification process yields a number of designs which match the user's specifications.

These designs are used as starting points for further optimization (Chapter 4). Optimization is implemented using a non-linear programming algorithm called *donlp2*. This process requires function values and derivatives at every point in the design space. Since our design space is not well characterized, we use a statistical function approximation tool called MARS (MultiVariate Adaptive Regression Splines) to obtain a model of the functional relationships between inputs and outputs. This function approximation is then used to provide us with the necessary derivatives. Concepts from Multi Objective optimization are also examined to perform a multiple objective optimization. We choose the TradeOff formulation, which attempts to optimize a primary attribute while keeping others within specified limits.

In Chapter 5 we analyze the NDA from the point of view of the software system and present a number of implementation details. The user interface of the NDA system is described, and other components supplied by our industrial sponsors are introduced. A comprehensive set of examples is tried using industrial solvers, and summary results from these experiments, and other analyses are presented in Chapter 6.

7.2 Conclusions and Thesis Contributions

The NDA has been very successful in meeting its objectives. The design sub-space identification process yields a number of solutions matching the user's requirements. This allows the user to perform a multiple scenario analysis before making a decision. Optimization capabilities are also built-in to help the designer optimize on a particular attribute (still keeping the others within specified limits). The entire process is completed in a few hours, as opposed to a few weeks spent on this process when this process is carried out in industry without the aid of this tool. This cuts down the development lead time drastically, and can allow a firm to react to customer requests faster.

In short, the NDA has met all of its objectives and has resulted in a viable software tool which is useful to the industry. This tool is currently customized for use within the design environment of our industrial sponsors. This is an evolution from our initial academic quest of developing a very general tool which would function without any user intervention or design environment constraints. This tool is more specific to the solvers incorporated herein but this customization has made it into a very powerful tool for an industrial design environment. The methodologies used, which have been described in this thesis, are sufficiently general and may be applied to a large class of problems.

This thesis' specific contributions may be summarized into three main points:

- Developed an integrated design methodology for design automation and optimization of complex engineering artifacts.

- Combined the power of a function approximation technique called MARS, and conventional optimization, to yield a very general and powerful optimization methodology.

- Developed an industrial strength software design tool for design automation and optimization.

7.3 Suggestions for Future Work

This thesis is complete in terms of fulfilling its objectives. However, like every other research and development project, there is always scope for improvement, and there is always a chance of exploring certain issues further. We will attempt to classify our suggestions into three categories: New Research Directions, Improvements to the NDA as a design tool, and Improvements to the User Interface.

New Research Directions

Research presented in this thesis opens up a number of possible research directions, almost all of which could be subjects for future work.

- Design Sub-Space algorithm. The design sub-space identification algorithm has worked very well for our purposes, both in [96] and in this thesis. However, it still represents only one possible way of navigating the design space. Future research could investigate some of the newer techniques in statistics related to Quasi Monte Carlo methods which attempt to ensure a better probabilistic coverage of the design space. This field is fairly new and has not found many engineering applications. Interested readers are referred to [52], [58], [74], and the numerous references therein to get started.

- Optimization. Optimization is a vast area of research. Numerous possibilities exist for extending the work presented in this thesis. We would like to refer the interested researcher to the numerous multi objective optimization concepts, and other novel optimization algorithms like simulated annealing and genetic algorithms. In fact, it would be worthwhile to do a comparison of Genetic Algorithms (or Simulated Annealing) to the techniques proposed in this thesis.

- MARS. MARS for multiple input multiple output (MIMO) systems is another interesting area of research. Currently, MARS only has Multiple Input Single Output (MISO) capabilities.

- Offer NDA like design tools with customizable components (like solvers) and have it available on the web for a large audience to use. Such a tool could have a user interface written in Java which makes it possible to run it on many different platforms.

Improvements to the NDA Design Tool

There are certain improvements which can be made to the NDA as a design tool which would enhance its capabilities. Most of these changes, however, are not trivial and are possible new research directions in their own right.

- Adding extra attributes. Currently, we work within a framework where the number of attributes is fixed for a particular solver. However, for certain cases, e.g. special purpose motors, some additional attributes may be of interest. It would be worthwhile to have the ability to add and delete attributes at will. This problem is pretty complex, since it raises issues about the generality of a software system. Most attributes need to be queried from the solver and hence, if a new attribute is added, it is very difficult to "add" a function which will query the solver for this new attribute. This is possible to do before the program is compiled but is extremely difficult while the program is running.

- Adding other solvers. Similarly, it would be helpful if an expert user could add an entire solver to the existing framework without modifying the code. This task may again prove to be near to impossible in the current form of the NDA but could be explored by future designers for different kinds of tools.

- Rule Sets. Currently, most of the rule sets used by the NDA are hardwired into the code. It would be interesting to have the capabilities for adding or deleting rules to be imposed on the variables. This task is again difficult because currently these rules have to be written and compiled into the program.

- Additional Optimization Algorithms. Many optimization algorithms could be candidates for an NDA like design tool. Even for the same class of problems (say the same solver), different optimization routines may yield different results depending on

the requirements and conditioning of the problem. An interesting research direction, and perhaps a thesis in its own right, is the facility of having multiple optimization algorithms to choose from. The system could automatically decide the algorithm to use depending on the problem specifications and conditioning of variables.

- Cost Modules and Error Estimates. As demonstrated in Chapter 6 adding a cost module, and error estimates, to solvers would enhance the decision making capabilities of the NDA tremendously.

Improvements to the User Interface

There are numerous possibilities for improving the user interface. [121] says that 90% of a commercial software's development time is spent in designing the user interface. Clearly, since our objectives were research oriented, we did not apportion 90% of our time to the user interface. Some suggestions are listed below. There may be numerous others which a user (or the reader) could suggest and add for other design tools.

- Have a facility of calling up earlier design processes and making modifications to them. This could include the possibility of repeating a previous run (such that the random number sequence is also the same).

- Have a facility in the tool to tell the user how much time it would take before a given task is completed (since some of the operations are very time consuming). The user could then decide whether they would want to go ahead with the operation.

- Letting the user work in parallel. Currently, when the system is performing an operation (which could be several hours), the user cannot do anything with the NDA screen which remains frozen. Modifications could be made which would allow a process to run in the background and let the user perform some other tasks in the NDA at the same time e.g. view previous designs, but not be able to start a process which might interfere with the existing process.

- Abort a process. A simple prelude to the above idea is let the user abort a run in the middle without having to kill the program (like a "Stop" button on a web browser).

Bibliography

[1] A. Agresti. *Categorical Data Analysis.* John Wiley and Sons – A Wiley-Interscience Publication, 1990.

[2] P. L. Alger. *The Nature of Induction Machines.* Gordon and Breach Science Publishers Inc., 150 Fifth Avenue, New York, NY 10011, 1965.

[3] O. W. Anderson. Optimum Design of Electrical Machines. *IEEE Transactions on Power Apparatus and Systems*, 86(6), June 1967. pp 707–711.

[4] J. Appelbaum, E. F. Fuchs, and J. C. White. Optimization of Three-Phase Induction Motor Design. *IEEE Transactions on Energy Conversion*, EC-2(3), September 1987. Part I pp 407-414, Part II pp 415-422.

[5] D. Bae, D. Kim, H. Jung, S. Hahn, and C. S. Koh. Determination of Induction Motor Parameters by Using Neural Network Based on FEM Results. *IEEE Transactions on Magnetics*, 33(2), March 1997.

[6] T. Bardasz and I. Zeid. Applying Analogical Problem Solving to Mechanical Design. *Computer Aided Design*, 23(3), April 1991. pp 202–212.

[7] L. A. Belfore II and A. A. Arkadan. Modeling Faulted Switched Reluctance Motors using Evolutionary Neural Networks. *20th International Conference on Industrial Electronics, Control and Instrumentation*, 2 of 3, September 1994. Bologna, Italy.

[8] D. G. Bharadwaj, K. Venkatesan, and R. B. Saxena. Experience with Direct and Indirect Search Methods Applied to Cage Induction Motor Design Optimization. *Electric Machines and Electromechanics*, 4(1), 1979. pp 85-93.

[9] E. Binaghi. A Fuzzy Logic Inference Model for a Rule–Based System in Medical Diagnosis. *Expert Systems*, 7(3), August 1990. pp 134–141.

[10] Y. M. M. Bishop, S. E. Fienberg, and P. W. Holland. *Discrete Multivariate Analysis.* The MIT Press, Cambridge, Massachusetts, 1975.

[11] J. A. Bland and G. P. Dawson. Tabu Search and Design Optimization. *Computer Aided Design*, 23(3), April 1991. pp 195–201.

[12] K. D. Boese, A. B. Kahng, and C. A. Tsao. Best-So-Far vs. Where You Are: New Perspectives on Simulated Annealing for CAD. *IEEE*, 1993. pp 78-83.

[13] R. A. Brealey and S. C. Myers. *Principle of Corporate Finance.* McGraw Hill, New York, fourth edition, 1991.

[14] L. Breiman. The π Method for Estimating Multivariate Functions from Noisy Data. *Technometrics*, 33(2), May 1991. pp 125-160.

[15] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, Cole Advanced Books & Software, Pacific Grove, California, 1984.

[16] D. C. Brown. Failure Handling in a Design Expert System. *Computer Aided Design*, 17(9), 1985. pp 436–442.

[17] S. J. Chalmers and B. J. Bennington. Digital–Computer Program for Design Synthesis of Large Squirrel–Cage Induction Motors. *Proceedings of the IEE*, 114(2), February 1967. pp 261–268.

[18] Special Report Soft Computing. Fuzzy, Neural and Genetic Methods Train to Overcome Complexity. *Computer Design*, May 1995. pp 62–76.

[19] P. A. Crosby. Application of a Monte Carlo Optimization Technique to a Cluster of Wind Turbines. *Transactions of the ASME*, 109, November 1987. pp 330-336.

[20] F. Cuadra, J. J. Leon, and G. Solari. Application of Knowledge Engineering to a Complex Optimization Problem: The Allocation of Payloads in the Hermes Cargo Bay. *Artificial Intelligence and Knowledge–Based Systems for Space*, Workshop 22–24 May 1991. ESTEC, Noordwijk, The Netherlands.

[21] F. Cuadra, I. J. Perez-Arriaga, and J. H. Lang. The Application of Knowledge Engineering to the Computer Aided Design of Optimal Electric Drives. *Proceedings of the 1990 International Conference on Electrical Machines*, August 1990. pp 842–848.

[22] L. Davis ed. *Genetic Algorithms and Simulated Annealing.* Morgan Kaufmann Publishers Inc., 95 First Street, Los Altos, CA 94022, 1987.

[23] C. de Boor. *A Practical Guide to Splines.* Vol 27, Applied Mathematical Sciences: Springer-Verlag, 1978.

[24] R. D. De Veaux, D. C. Psichogios, and L. H. Ungar. A Comparison of Two Nonparametric Estimation Schemes: MARS and Neural Networks. *Computers in Chemical Engineering*, 17(8), 1993. pp 819-837.

[25] J. G. Doheny and P. F. Monaghan. IDABES: An expert system for the preliminary stages of conceptual design of building energy systems. *Artificial Intelligence in Engineering*, 2(2), 1987. pp 54–64.

[26] A. Duffy. Bibliography – Artificial Intelligence in Design. *Artificial Intelligence in Engineering*, 2(3), July 1987. pp 173–179.

[27] M. G. Dyer, M. Flowers, and J. Hodges. EDISON: An engineering design invention system operating naively. *Artificial Intelligence in Engineering*, 1(1), 1986. pp 36–44.

[28] S. M. Ervin and M. D. Gross. RoadLab – A Constraint Based Laboratory for Road Design. *Artificial Intelligence in Design*, 2(4), 1987. pp 224–234.

[29] H. Eschenauer, J. Koski, and A. (Eds) Osyczka. *Multicriteria Design Optimization - Procedures and Applications.* Springer Verlag, Berlin, Heidelberg, 1990.

[30] K. S. Eshbaugh. Generation of Correlated Parameters for Statistical Circuit Simulation. *IEEE Transactions on Computer Aided Design*, 11(10), October 1992. pp 1198-1206.

[31] R. L. Eubank. *Spline Smoothing and Nonparametric Regression.* Marcel Dekker Inc., 270 Madison Avenue, New York, NY 10016, 1988.

[32] M. J. Fagan. Expert Systems Applied to Mechanical Engineering Design – Experience with Bearing Selection and Application Program. *Computer Aided Design*, 19(7), September 1987.

[33] A. Farnham. Baldor's Success: Made in the USA. *Fortune*, July 1989. pp 101–104.

[34] N. H. Fetih and H. M. El-Shewy. Induction Motor Optimum Design, Including Active Power Loss Effect. *IEEE Transactions on Energy Conversion*, EC-1(3), September 1986. pp 155-160.

[35] A. E. Fitzgerald, C. Kingsley Jr., and S. D. Umans. *Electric Machinery.* McGraw–Hill Inc., 1221 Avenue of the Americas, New York, NY 10020, fifth edition, 1990.

[36] Open Software Foundation. *OSF/MOTIF Programmer's Guide.* Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, 1990.

[37] Open Software Foundation. *OSF/MOTIF Programmer's Reference.* Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, 1990.

[38] Open Software Foundation. *OSF/MOTIF Style Guide.* Open Software Foundation, 11 Cambridge Center, Cambridge, MA 02142, 1990.

[39] J. H. Friedman. Fitting Functions to Noisy Data in High Dimensions. *Proceedings of the Twentieth Symposium on the Interface, Wegman, Gantz, and Miller, eds. American Statistical Association*, 1988. Alexandria, VA. pp 3-43.

[40] J. H. Friedman. Estimating Functions of Mixed Ordinal and Categorical Variables using Adaptive Splines. *Department of Statistics, Stanford University*, 1991. Technical Report LCS108.

[41] J. H. Friedman. Multivariate Adaptive Regression Splines (with discussion). *Annals of Statistics*, 19(1), March 1991. pp 1-141.

[42] J. H. Friedman and B. W. Silverman. Flexible Parsimonious Smoothing and Additive Modeling (with discussion). *Technometrics*, 31, February 1989. pp 3-39.

[43] J. H. Friedman and W. Stuetzle. Projection Pursuit Regression. *Journal of the American Statistical Association*, 76, 1981. pp 817-23.

[44] S. B. Gelfand and S. K. Mitter. Simulated Annealing Type Algorithms for Multivariate Optimization. *Laboratory for Information and Decision Systems, Massachusetts Institute of Technology*, Technical Report(LIDS-P-1845), January 1989.

[45] S. B. Gelfand and S. K. Mitter. Metropolis-type Annealing Algorithms for Global Optimization in Rd. *Laboratory for Information and Decision Systems, Massachusetts Institute of Technology*, Technical Report(LIDS-P-1977), May 1990.

[46] J. S. Gero. Bibliography of books on Artificial Intelligence with Particular Reference to Expert Systems and Knowledge Engineering. *Computer Aided Design. Special Issue: Expert Systems*, 17(9), November 1985.

[47] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, Massachusetts, 1989.

[48] T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models.* Chapman and Hall, New York, 1990.

[49] P. G. Hoel. *Introduction to Mathematical Statistics.* John Wiley and Sons Inc., New York, fifth edition, 1984.

[50] R. V. Hogg and J. Ledolter. *Applied Statistics for Engineers and Physical Scientists.* Macmillan, New York : Toronto : New York : Macmillan ; Maxwell Macmillan; Maxwell Macmillan International, 2nd edition, 1992.

[51] A. E. Howe, P. R. Cohen, J. R. Dixon, and M.K. Simmons. Dominic: A Domain Independent Program for Mechanical Engineering Design. *Artificial Intelligence in Engineering*, 1(1), 1986. pp 23–28.

[52] R. L. Iman. Uncertainty and Sensitivity Analysis for Computer Modeling Applications. *ASME Transactions on Reliability Technology*, AD-28, 1992. pp 153-168.

[53] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis.* Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 3rd edition, 1992.

[54] Y. E. Kalay. Redifining the role of computers in architecture: from drafting/modelling tools to knowledge–based design assistants. *Computer Aided Design*, 17(7), September 1985. pp 319–328.

[55] B. W. Kernighan and R. Pike. *The Unix Programming Environment.* Prentice–Hall of India Private Limited, New Delhi, India, 1989.

[56] B. W. Kernighan and D. M. Ritchie. *The C Programming Language.* Prentice–Hall of India Private Limited, New Delhi, India, second edition, 1990.

[57] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598), 13 May 1983. pp 671-680.

[58] J. R. Koehler and A. B. Owen. Computer Experiments. *Department of Statistics, Stanford University*, 1995. Technical Report.

[59] E. Kreyszig. *Advanced Engineering Mathematics.* Wiley Eastern Limited, 4835/24 Ansari Road, Daryaganj, New Delhi, India 110002, fifth edition, 1983.

[60] R. Krishnan, A. S. Bharadwaj, and P. N. Materu. Computer Aided Design of Electrical Machines for Variable Speed Applications. *IEEE Transactions on Industrial Electronics*, 35, November 1988. pp 560–571.

[61] C. F. Landy, R. Kaplan, and V. Lun. An Expert System for the Design of 3–Phase Squirrel Cage Induction Motors. *IEE Conference Proceedings. Third International Conference on Electrical Machines and Drives, London*, 1987.

[62] C. C. Lee. Fuzzy Logic in Control Systems: Fuzzy Logic Controller–Part I. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2), March/April 1990. pp 404–417.

[63] K. S. Leung, W. S. Felix Wong, and W. Lam. Applications of a Novel Fuzzy Expert System Shell. *Expert Systems*, 6(1), February 1989. pp 2–10.

[64] P. A. W. Lewis and J. G. Stevens. Nonlinear Modeling of Time Series Using Multivariate Adaptive Regression Splines (MARS). *Journal of the American Statistical Association*, 86(416), December 1991. pp 864-877.

[65] R. P. Lippmann. An Introduction to Computing with Neural Nets. *IEEE-ASP Magazine*, April 1987. pp 4–22.

[66] G. Madescu, I. Boldea, and T. J. E. Miller. An Analytical Iterative Model (AIM) for Induction Motor Design. *SPEED Report, University of Glasgow*, September 1996.

[67] G. Madescu, I. Boldea, and T. J. E. Miller. The Optimal Lamination Approach (OLA) to Induction Machine Design Global Optimization. *SPEED Report, University of Glasgow*, September 1996.

[68] M. L. Maher. HI–RISE and Beyond: Directions for Expert Systems in Design. *Computer Aided Design*, 17(9), November 1985. pp 420–427.

[69] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis.* Academic Press Inc., 24/28 Oval Road, London NW1, 1979.

[70] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall, 29 West 35th Street, New York, NY 10001, second edition, 1989.

[71] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21(6), June 1953. pp 1087-1092.

[72] J. A. Moses. A Design Assistant for Induction Motors. *S. M. Thesis, Massachusetts Institute of Technology*, September 1991.

[73] J. A. Moses, J. L. Kirtley Jr, J. H. Lang, R. D. Tabors, and F. de Cuadra. A Computer Based Design Assistant for Induction Motors. *Conference Record of the 1991 IEEE IAS Annual Meeting, Dearborn MI*, October 1991.

[74] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, siam 63 edition, 1992.

[75] J. W. Nims, R. E. Smith, and A. A. El-Keib. Application of a Genetic Algorithm to Power Transformer Design. *Electric Machines and Power Systems*, 24(6), 1996. pp 669-680.

[76] R. Nolan, P. Pillay, and T. Haque. Application of Genetic Algorithms to Motor Parameter Determination. *EMC*, 21, 1994.

[77] A. Osyczka. *MultiCriterion Optimization in Engineering.* Ellis Horwood Limited (a division of John Wiley & Sons), Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England, 1984.

[78] R. Oxman and J. S. Gero. Using an Expert System for Design Diagnosis and Design Synthesis. *Expert Systems: The International Journal of Knowledge Engineering*, 4(1), February 1987.

[79] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C - The Art of Scientific Computing.* Cambridge University Press, 40 West 20th Street, New York, NY 10011–4211, 1991.

[80] A. D. Radford and J. S. Gero. Towards Generative Expert Systems for Architectural Detailing. *Computer Aided Design*, 17(9), November 1985. pp 428–435.

[81] S. Rai. Computer–Aided Structure Redesign of High Speed Electromechanical Systems for Improved Control. *PhD Thesis, Massachusetts Institute of Technology*, May 1993.

[82] R. Ramarathnam and B. G. Desai. Optimization of Polyphase Induction Motor Design: A Non-Linear Programming Approach. *IEEE Transactions on Power Apparatus and Systems*, PAS-90(2), March/April 1971.

[83] D. R. Rehak and H. C. Howard. Interfacing Expert Systems with Design Databases in Integrated CAD Systems. *Computer Aided Design*, 17(9), 1985.

[84] M. Richeldi. Improving the Effectiveness of the Knowledge Discovery Process by Detecting Higher-Order Correlations Between Data. *European Conference on Machine Learning - Workshop on Statistics in Machine Learning*, 23(1), February 1995. pp 15-24.

[85] M. A. Rosenman and J. S. Gero. Design Codes as Expert Systems. *Computer Aided Design*, 17(9), 1985. pp 399–409.

[86] M. A. Rosenman, J. S. Gero, P. J. Hutchinson, and Oxman R. Expert Systems Applications in Computer Aided Design. *Computer Aided Design*, 18(10), December 1986. pp 546–551.

[87] M. D. Rychener. Expert Systems for Engineering Design. *Expert Systems: The International Journal of Knowledge Engineering*, 2(1), January 1985.

[88] Jung W. S. and N. Z. Cho. Determination of Design Alternatives and Performance Criteria for Safety Systems in a Nuclear Power Plant via Simulated Annealing. *Reliability Engineering and System Safety*, 41, 1993. pp 71-94.

[89] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization.* Academic Press, Orlando, Florida 32887, 1985.

[90] S. Sekulic and B. R. Kowalski. MARS: A Tutorial. *Journal of Chemometrics*, 6, 1992. pp 199-216.

[91] P. Serafini Ed. *Mathematics of Multi Objective Optimization.* Springer-Verlag, Wien-New York, cism 289 edition, 1985.

[92] V. Sharma. A New Modeling Methodology Combining Engineering and Statistical Modeling Methods: A Semiconductor Manufacturing Application. *ScD Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology*, September 1996.

[93] B. Singh, B. P. Singh, S. S. Murthy, and C. S. Jha. Experience in Design Optimization of Induction Motor using SUMT Algorithm. *IEEE Transactions on Power Apparatus and Systems*, 102(10), October 1983.

[94] G. Singh, S. C. Srivastava, P. K. Kalra, and D. M. Vinod Kumar. Fast Approach to Artificial Neural Network Training and its Application to Economic Load Dispatch. *Electric Machines and Power Systems*, 23, 1995. pp 13–24.

[95] U. Sinha. Design and Control of a Flexible Manufacturing System using Computer Simulation and Expert System Approach. *B–Tech Thesis, Indian Institute of Technology, New Delhi, India*, May 1991.

[96] U. Sinha. A Design Assistant for Induction Motors. *S.M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology*, August 1993.

[97] U. Sinha and J. L. Kirtley Jr. Design Synthesis of Induction Motors. *International Conference on Electric Machinery*, September 1994. Paris, France.

[98] K. Srinivasan. Integrated Design of High-Speed Permanent-Magnet Synchronous Motor Drives. *PhD Thesis, Massachusetts Institute of Technology*, November 1995.

[99] D. Sriram. ALL–RISE: A Case Study in Constraint–Based Design. *Artificial Intelligence in Design*, 2(4), 1987. pp 186–203.

[100] W. Stadler Ed. *MultiCriteria Optimization in Engineering and in the Sciences.* Plenum Press, 233 Spring Street, New York, NY 10013, 1988.

[101] N. P. Suh. *The Principles of Design.* Oxford University Press, 1990.

[102] N. P. Suh. Axiomatic Design of Mechanical Systems. *A special combined issue of J. Mechanical Design and J. Control, Transactions of the ASME*, 117, June 1995.

[103] N. P. Suh. Design and Operation of Large Systems. *Journal of Manufacturing Systems*, 14(3), 1995. pp 203–213.

[104] N. P. Suh. Designing-in of Quality Through Axiomatic Design. *IEEE Transactions on Reliability*, 44(2), June 1995. pp 256–264.

[105] N. K. Taylor and E. N. Corlett. An expert system which constrains designs. *Artificial Intelligence in Engineering*, 2(2), 1987. pp 72–75.

[106] T. V. Theodosopoulos. Review of Optimal Monotone Annealing Schedules for Global Optimization. *Laboratory for Information and Decision Systems, Massachusetts Institute of Technology*, Technical Report(LIDS-P-2229), February 1994.

[107] T. V. Theodosopoulos. Stochastic Models for Global Optimization. *PhD Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, May 1995.

[108] M. Tolikas. The Application of Homotopy Methods in the Analysis of Electric Power Systems. *S.M. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, 1992.

[109] M. Tolikas. Dual-Energy Electromagnetic Modeling, with Application to Variable Reluctance Motor Analysis. *PhD Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, August 1995.

[110] J. N. Tsitsiklis. A Survey of Large Time Asymptotics of Simulated Annealing Algorithms. *Laboratory for Information and Decision Systems, Massachusetts Institute of Technology*, Technical Report(LIDS-P-1623), November 1986.

[111] C. L. Tucci. Incorporating Manufacturing and Cost Accounting into the Design Process of Induction Motors. *S. M. Thesis, Massachusetts Institute of Technology*, September 1991.

[112] C. L. Tucci, J. H. Lang, R. D. Tabors, and J. L. Kirtley Jr. A Simulator of the Manufacturing of Induction Motors. *Conference Record of the IEEE IAS Annual Meeting, Dearborn, MI*, October 1991.

[113] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Company, P.O. Box 17, 3300 AA Dordrecht, Holland, 1987.

[114] D. Vanderbilt and S. G. Louie. A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables. *Journal of Computational Physics*, 56, 1984. pp 259-271.

[115] C. G. Veinott. *Theory and Design of Small Induction Machines.* McGraw Hill, 1959.

[116] C. G. Veinott. *Computer Aided Design of Electric Machinery.* MIT Press, Cambridge, MA, 1972.

[117] M. B. Wall. A Genetic Algorithm for Resource-Constrained Scheduling. *PhD Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology*, May 1996.

[118] D. R. Wallace, M. J. Jakiela, and W. C. Flowers. Design Search under Probabilistic Specifications using Genetic Algorithms. *Computer-Aided Design*, 28(5), 1996. pp 405–421.

[119] S. Williamson and C. I. McClay. Optimization of the Geometry of Closed Rotor Slots for Cage Induction Motors. *IEEE Transactions on Industry Applications*, 32(3), May-June 1996. pp 560-568.

[120] P. H. Winston. *Artificial Intelligence.* Addison–Wesley Publishing Company, Reading, Massachusetts, third edition, 1992.

[121] D. A. Young. *The X Window System - Programming and Applications with Xt.* Prentice Hall Inc., Englewood Cliffs, NJ 07632, osf/motif edition, 1990.