PRACTICAL AND ANALYTICAL STUDIES ON THE DEVELOPMENT OF FORMAL EVALUATION AND DESIGN METHODOLOGIES FOR MECHATRONIC SYSTEMS

by

SAEED BEHBAHANI

B.Sc. (Hons.), Mechanical Engineering, Amir Kabir University of Technology, Tehran, Iran, 1996
M.Sc. (Hons.), Mechanical Engineering, Amir Kabir University of Technology, Tehran, Iran, 1998

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
February 2007

Abstract

The integration of mechanical engineering, electrical engineering and information technology in one mixed system has found vast applications in industry and everyday life. This interdisciplinary field, known as Mechatronics, has attracted a great deal of attention, particularly in the context of optimal design of multi-domain systems. To this end, the present thesis represents an original investigation into the development of formal and systematic methodologies for the optimal design and design evaluation of mechatronic systems.

This work presents a new philosophy and approach for the optimal design of mechatronic systems. It takes into account that an optimal mechatronic design requires concurrent, integrated, and system-based thinking with regard to all design parameters and criteria involved in the mechatronic system. The mathematical model and axioms which support this thinking are presented in the thesis. A design evaluation index has been presented in the present work which supports above statement. It is based on the concept of mechatronic design quotient (MDQ). MDQ is a multi-criteria index reflecting the overall degree of satisfaction of the design criteria for a mechatronic system. In this thesis, a nonlinear fuzzy integral is used for the aggregation of the various design criteria and for handling possible correlations among them. For an existing mechatronic system, MDQ is a useful index for evaluating its design as well, and determining the potential for improvement. In different stages of design, it can be used as an index for the purposes of optimization and/or decision making.

In the present work, a new systematic mechatronic design methodology based on the concept of MDQ maximization is presented. The design procedure is treated in multiple stages. In the conceptual stage, MDQ provides guidance to the designer in selecting the best design choices and making effective decisions about the essential structure of the design. In the next stage, which concerns detailed design, a niching genetic algorithm is employed to find the elite design alternatives for all possible configurations and combinations of system parts. Since a full MDQ assessment is computationally expensive, it is not practical to consider all MDQ attributes in the course of an evolutionary optimization. Only the essential criteria which have a veto effect on the MDQ evaluation are considered in the process of the niching genetic algorithm. A full and detailed MDQ assessment is then employed to find the best choice among the elite representatives.

The reliability assessment of mechatronic systems is studied as well in the thesis. A new reliability assessment methodology is developed which has two practical advantages over the available methodologies. First, in view of dynamic interactions that may exist in a mechatronic system, the developed method uses a Petri-net simulation of the dynamic behavior of the system. Here, all possible events and conditions of the real operation of the system and all possible interactions can be accurately modeled in the level of detail that the designer prefers. Second, the severity of the failure modes is considered in the reliability evaluation methodology. The developed reliability evaluation approach contributes in the mechatronic design process by revealing information about the performance of various design choices.

A bond graph mechatronic simulation tool is developed in this work. It is based on a new matrix-based formulation, which is presented in the thesis. The bond graph simulation tool is integrated with genetic programming to form a unified evolutionary mechatronic tool. As a new contribution, this synergic integration is extended for the general case of nonlinear mechatronic problems. This tool can concurrently optimize both the topology and "size" of a bond graph model of a mechatronic system in order to achieve the best fitness for the solution. It can be used in any mechatronic problem, provided that an effective fitness evaluation is established for the problem. In particular, the application of this tool for automated system identification of nonlinear mechatronic systems is presented in the thesis.

The methodologies developed in this research are validated by applying them to the modeling and redesign process of an industrial fish processing machine, called the Iron Butcher – a complex electromechanical system which falls into the class of mixed or multi-domain systems.

Table of Contents

List of Tables

List of Figures

Nomenclature

A	A set of criteria; piston area in hydraulic cylinder
a	Weighting factor
B	A set of criteria
b	Weighting factor
C	Capacitor
c_1	Slope in Stribeck region of the friction model.
c_2	Slope after Stribeck region of the friction model.
C_{eq}	Equivalent capacitor
$c_{eq}(X)$	A behavior specification which needs to be equal to a specified value
$c_{max}(X)$	A behavior specification to be maximized
$c_{min}(X)$	A behavior specification to be minimized
d	Backlash
E	Energy; module of elasticity
e	Effort variable
F	Cumulative distribution of the population fraction
F_f	Friction magnitude
F_{mat}	Force matrix
F_s	Static friction
f	Flow Variable; fitness of a trial solution in evolutionary algorithms
$\tilde{f}_i(t)$	Partial failure value of the i^{th} machine in the time t
G	An operator indicating whether a constraint has been met
$g_e(X)$	Equality constraint
$g_i(X)$	Inequality constraint
H	Aggregation operator; the set of inhibitor arcs from places to transitions in a Petri net
I	Inductance; the set of directed arcs from places to transitions in a Petri net
I_e	Electrical inductance
I_m	Mechanical inductance

I_{eq}	Equivalent Inductance
k_b	The stiffness of the spring used to simulate backlash
K_{mat}	Stiffness matrix
M	Number of comparison points between actual response and desired response for fitness calculation
MDQ	Mechatronic design quotient
m	Number of design variables; the set of markings in a Petri net
m_0	Initial marking in a Petri net
m'	Marking of a Petri net after firing of a transition
N	The population of a generation in evolutionary algorithms; the initial number of machines in a reliability analysis
$\mathbb{N}$	The set of natural numbers
NF	Number of failure modes
n	Number of criteria; the rank of the selected individual in evolutionary algorithms
n_f	Number of failed machines
n_h	Number of machines still operating
n_{ij}	Number of times that the j^{th} failure mode has happened in the i^{th} sample machine
$\tilde{n}_j$	Number of times a failure mode has occurred
$\tilde{N}_j$	Total number of times that occurrence of a failure mode has been examined.
O	Occurrence probability of a failure mode; set of directed arcs from transitions to places in a Petri net
P	Pressure; probability of selection of an individual in evolutionary algorithm; a finite set of places in a Petri net
P_1, P_2	Output pressures from servo valve
p	Momentum variable; a place in a Petri net
q	Displacement variable; flow rate
R	Resistance; reliability
R_{eq}	Equivalent resistance
R_e	Electrical resistance
R_m	Mechanical resistance
r	Number of constraints; a random number
r_G	Gyrator modulus

r_T	Transformer modulus
R_{eq}	Equivalent resistance
S	Aggregated score; a trial solution in evolutionary algorithms; motion stroke in hydraulic cylinder; Severity of a failure mode
S_e	Source of effort
$S_{e_{eq}}$	Equivalent source of effort
S_f	Source of flow
$S_{f_{eq}}$	Equivalent source of flow
s_i	The partial score of a design choice against criterion y_i
T	A set of criteria; a finite set of transitions in a Petri net
t	Time; a transition in a Petri net
v	Velocity response of the model
v'	Velocity response of the actual system
$\bar{v}$	The mean value of the absolute values of the actual velocity response
V_c	Velocity margin in Stribeck friction model
V_L	Volume of the connected pipes to hydraulic cylinder
X	Set of design variables
X_d	Set of desired design variables
x	Displacement response of the model
x'	Displacement response of the actual system
$\bar{x}$	The mean value of the absolute values of the actual displacement response
x_i	i^{th} design variable
Y	The universe set of behavior specifications
Y_d	Set of desired behavior specifications
y_i	i^{th} criterion (behavior specification)
Z	A Petri net
α	Characteristic life
β	Shape parameter.
ε	The numeric value resulted from an arithmetic tree in GP
ϕ	Null set
γ	Starting time for the life of a component

λ	Failure probability
μ	A fuzzy measure indicating degree of importance of a set of criteria
$\mu(i,j)$	Fuzzy measure of the set of criteria which consists only y_i and y_j
Θ	Interaction index
ρ	Modification factor
Ω	Specified value for a behavior specification
ω	A typical parameter of the response of the model of a system used for fitness calculation
ω'	A typical parameter of the desired response of a system used for fitness calculation
$\overline{\omega}$	The average of absolute values of the response.
ξ	Discrimination factor in evolutionary algorithms
Ψ	Overall importance index (Shapley value)
ζ	A variable to be optimized by evolutionary algorithms
ζ_d	The default value for the parameter ζ

Abbreviations

ABS	Antilock Brake System
BG	Bond Graphs
C	Capacitor
CAMPG	Computer Aided Modeling Program with Graphical input
C-I	Compound-Individual
CMS	Component Mode Synthesis
DC	Deterministic Crowding
DDC	Dispersing Deterministic Crowding
DFC	Design For Control
ESPN	Extended Stochastic Petri-Net
FE	Finite Elements
FMEA	Failure Modes and Effect Analysis
FTA	Fault Tree Analysis
GA	Genetic Algorithm
GP	Genetic Programming
GSPN	Generalized Stochastic Petri Nets
GY	Gyrator
I	Inductance
ITG	Intelligent Task Graph
MDOF	Mechanical Degree of Freedom
MDQ	Mechatronic Design Quotient
MIQ	Machine Intelligent Quotient
MRS	Modular Robotic System
NonRTB	Non Real Time Behaviors
NonRTP	Non Real Time Parameters
OOM	Object Oriented Modeling
P	Proportional controller
PD	Proportional and Derivative controller
PDF	Probability Density Function

PI	Proportional and Integrator controller
PID	Proportional, Integrator, and Derivative controller
PN	Petri Nets
PSM	Pattern Search Method
R	Resistance
RCS	Restricted Competition Selection
RPN	Risk Priority Number
RTB	Real Time Behaviors
RTP	Real Time Parameters
RTS	Restricted Tournament Selection
S-C	Sub-Concept
SPN	Stochastic Petri Nets
TBF	Time Between Failures
TF	Transformer
TTFF	Time To First Failure

Acknowledgments

I wish to express my sincere gratitude to my supervisor, Prof. Clarence W. de Silva for his strong support, perfect guidance, generous advice, fine teaching, invaluable patience, and proactive attitudes toward innovative ideas. I thank him for taking a very active interest in my work, his patience in editing all my publications and reports, and persistent encouragement to keep me motivated towards my research and academic goals. I am greatly indebted to him for countless opportunities he provided for me, such as accepting to supervise me, providing financial support for my research, funding me to attend conferences and workshops, providing classroom teaching opportunities, and recommendation for scholarships.

Funding of my research has been provided from several generous sources. These include the Ministry of Science, Research, and Technology (MSRT) of Iran, through my supervisor from the National Sciences and Engineering Research Council (NSERC) of Canada, Ph.D. Tuition Fee Awards, and University Graduate Fellowships (UGF).

I would also like to thank the members of my supervisory committee, Prof. Farrokh Sassani and Prof. Elizabeth Croft from the University of British Columbia and Prof. Lalith Gamage of the Sri Lanka Institute for Information Technology, for their constructive comments and suggestions. I am indebted to other UBC faculty members; particularly Prof. Yusuf Altintas, Prof. Ian Yellowley, Prof. Derek Yip-Hoi, Prof. Mu Chiao, Prof. Robert Rohling, and Prof. Joseph Yan for their careful and valuable comments and suggestions on our annual graduate seminars, and/or for constructive knowledge they gave me through courses which they taught. I thank all of them.

Thanks also go to my friends and colleagues in the Industrial Automation Laboratory of UBC, under the directorship of Prof. Clarence de Silva; particularly, Dr. Poi Loon Tang, Mr. Ying Wang, Mr. Tao Fan, Mr. Richard McCourt, Mr. Duminda Wijewardene, Mr. M. Tahir Khan, and other members of my research group, particularly Mr. Jian Jason Zhang, Mr. Mohammad Alrasheed, Ms. Nazli Pirmoradi, Dr. Ken Wong, and Dr. Farbod Khoshnoud. I thank all of them for their help and making the lab a great and friendly environment.

As a personal note, I would like to thank my beloved family; particularly my kind wife, children and parents for their continuous love, support, and encouragement throughout my life. ***I dedicate this thesis to them.***

Foremost, I sincerely thank god, who gives me power to overcome the barriers to reach my goals, and who put all these fine people in my life.

Chapter 1

Introduction

In some literature, Mechatronics is defined as the synergetic combination of precision engineering, electronics, control technology, and system thinking in the design of products and processes (Van Brussel, 1996). In some others, Mechatronics is viewed as the synergistic application of mechanics, electronics, control engineering, and computer science in the development of electromechanical products and systems, through integrated design (De Silva, 2005). The general concept is shown in Figure 1.1.

The term Mechatronics can be traced back to the late 1960s, but it is primarily in the 1990s that a significant growth of the field was seen with regard to "mechatronic" developments of electro-mechanical systems. Regardless of the definition and whether it is just a new term or a new engineering field, it has received significant exposure recently due to widespread application of mixed systems in industry and in everyday life. Machine tools, biomechanics, robotics, automobiles, aerospace systems, food processing machinery, and home appliances are some examples where mechatronic systems and technologies have been incorporated with desirable results.

Due to widespread application of mechatronic systems and the competition to offer better

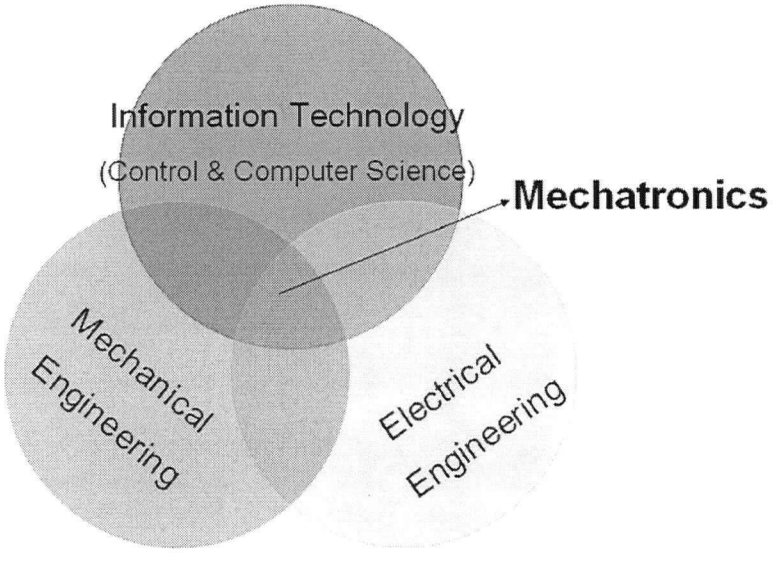


Figure 1.1: General representation of Mechatronics.

mechatronic products at lower cost, there has been a renewed attention in the area of optimal design. It is easy to see that traditional sequential design methods are not optimal. In this context, it has been shown that a concurrent and integrated design methodology is more optimal than a sequential design, from the mechatronic viewpoint (Zhang *et al.,* 1999; Li *et al.,* 2001; De Silva, 2003; De Silva, 2005). This emphasizes a clear difference between a mechatronic design, which is synergistic and concurrent, and a traditional electro-mechanical design, which is sequential. The motivations for concurrent and integrated mechatronic design include the following:

- Increased efficiency
- Cost effectiveness
- Ease of system integration
- Ease of cooperation with other systems
- Better component matching
- Increased reliability

Although by definition a concurrent and integrated design approach is required for a mechatronic system, design engineers still continue to follow traditional and sequential design methods. The main reason for this is the lack of a formal systematic and computationally manageable approach for mechatronic design.

1.1 Mechatronic Design Model

An engineering "parametric" design can be considered as a mapping from a required behavior space to a parameter space without violating a specified set of constraints. For a mechatronic design, Li *et al.*, (2001) proposed to the divide requirement space into two subspaces (Figure 1.2), which separately represent:

- Real-time behaviors (RTB)
- Non-real-time behaviors (NonRTB)

Following this division, the system parameters in the structural space can also be divided into two subspaces (Li *et al.*, 2001), which separately represent:

- Real-time and controllable parameters (RTP)
- Non-real-time and uncontrollable parameters (NonRTP).

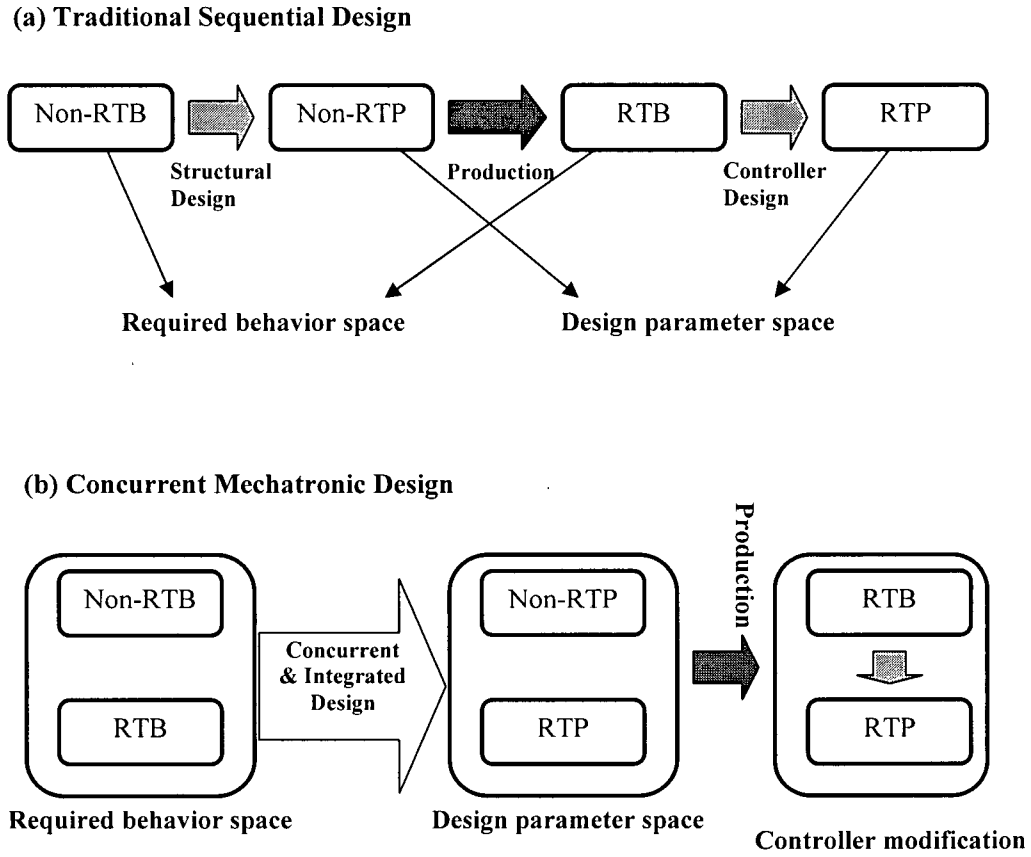


Figure 1.2: Model of mechatronic design. (a) Sequential design; (b) Concurrent design.

Here real-time denotes parameters and specifications which may change with time after the machine is built; for example, controller gains, accuracy, and speed. In contrast, non-real-time denotes parameters and specifications which may not change with time after the machine is built; for example, structural material, weight, and workspace.

In a traditional sequential design process (termed electromechanical design), NonRTP are designed first based on the NonRTB specifications. This process itself is performed in two stages:

1-The mechanical structure of the machine is designed (e.g., deciding on the configuration of the machine, sizing of the structure, selection of the structural material) according to some NonRTB specifications (e.g., allowable stresses, allowable workspace).

2-Electrical components are added to the machine (e.g., actuators, sensors, amplifiers, filters) based on the NonRTB requirements (e.g., maximum required acceleration, minimum acceptable accuracy).

Subsequently, RTP (e.g., control algorithm, controller gain, signal conditioning algorithm) are determined based on RTB specifications (e.g., desired path, speed, accuracy, stability) to control the already-established structure.

In such a traditional design methodology, it would be costly and difficult to vary a parameter in the mechanical structure or change an electrical component after the machine is built, except in a repair stage. Therefore all parameters related to mechanical structure and electrical hardware are hardly varied and can be stated as time invariant (NonRTP). In contrast, the parameters in the driving and control structure (strictly speaking, "signals") are time varying and by and large changeable (RTP), leading to the programmability of the electromechanical system. Recently, evolution of control engineering and computer science has resulted in the creation of a school of thought that the design of mechanical structure and electrical hardware are no longer the main design aspects in some electromechanical systems, and inadequacies of the system mechanics could be compensated for by sophisticated control schemes and software. This thinking can be self defeatist because a perfect control action may be hardly achieved due to hardware limitations and dynamic interactions, regardless of the effort devoted to the design of the control structure. This does not mean, however, that the performance of a machine cannot be improved by better control. It is the "adequacy" and the "optimality" that are in question.

In the research activity undertaken in the present thesis, it is believed that controllability and programmability of RTP should be considered as an opportunity to further improve the design after the machine is built, unlike in the traditional electromechanical design where programmability of RTP is considered as an excuse to postpone their design after the machine is built (Figure 1.2).

An ideal mechatronic design should also be system-based. It is clear that there are different criteria involved in mechatronic design. These criteria can have interactions with each other. Some criteria may have negative correlation with other criteria, which means high satisfaction of one of them implies low satisfaction of the other. A system-based design attempts to improve the product performance from different views considered together. In other words, it tries to improve the overall degree of satisfaction of the system objectives, which is an intuitive aggregation of different and sometimes opposite objectives. This implies that mechatronic design can be treated as a multi-objective problem.

The desired behavior of a system is presented by some design criteria. In general, the behavior of the system is a function of the system parameters (design variables):

$$Y = f(X) \tag{1.1}$$

where $X = [x_1, x_2, \ldots, x_m]$ is the set of design variables or decision variables and $Y = [y_1, y_2, \ldots, y_n]$ is the vector of behavior specifications or design criteria. The design process can then be considered as finding X_d so as to satisfy the required behavior specifications Y_d, without violating the constraints:

$$Y_d = f(X_d) \tag{1.2}$$

Desired behavior can be expressed in three basic ways:

- Maximizing a function $y_d \equiv Maximize[c_{\max}(X)]$
- Minimizing a function $y_d \equiv Minimize[c_{\min}(X)]$
- Satisfying an equality $y_d \equiv [c_{eq}(X) = \Omega]$

Two kinds of constraints may exist:

- Inequality constraints: $g_i(X) > 0$, and
- Equality constraints: $g_e(X) = 0$

If the physics of the problem gives a direct and explicit relation between the parameter space and the behavior space, then the desired parameters can be found simply by plugging the required behavior into the equations. For example, if the required safety factor and the maximum acceptable deflection of a beam are specified, its cross-section can be calculated by using standard elasticity equations. But in most engineering problems and particularly in mechatronic design, such a straightforward relation between design variables and behavior specifications cannot be found. Then an inverse design approach should be taken, where one starts from an embryo design, modifies the design variables, and makes proper decisions to achieve the desired behavior. Due to inequality expressions in the desired behavior, the solution to this problem may not be unique. The task of the designer is then to find the design parameters that not only satisfy the desired behavior without violating the constraints but also achieve the most desirable satisfaction.

1.2 Goals of the Research

Optimal mechatronic design needs an integrated, concurrent, and system-based approach. Machines that have been designed using traditional sequential design approaches, have the

potential for improvement through concurrent mechatronic design. The main challenges in the development of mechatronic design methodology are firstly to establish systematic and formal evaluation methods to formally represent the areas of potential improvement through concurrent design, and secondly to provide a systematic concurrent design method that can justifiably encourage designers to use the new methodology.

The first main objective of the research presented in this thesis is to develop a formal and systematic approach for the evaluation of mechatronic systems. The goal here is to provide an evaluation method with the following features:

- Rather general approach that covers a wide range of systems within the umbrella of mechatronic systems.
- Has the capability to incorporate past experience on mixed systems in the evaluation.
- A system-based approach that incorporates multiple criteria in the evaluation along with an intuitive aggregation method.

The second main objective is to develop a design flowchart for mechatronic design which provides a concurrent, integrated, and system-based method for achieving optimum mechatronic design.

The present thesis does not address the general problem of invention and innovation concerning products, processes, and systems. As such, general philosophies and concepts such as the theory of inventive problem solving (TIPS) or TRIZ in Russian (Altshuller, 1984) are not addressed in the thesis. The present research primarily focuses on the design and evaluation of "mechatronic" devices and systems, and the enhancement and application of the concept of Mechatronic Design Quotient or MDQ (De Silva, 2003) together with the development of tools for this purpose.

1.3 Related Work

Primarily the following four main research aspects have to be investigated in order to address the required background for the thesis, in the context of the indicated objectives:

- Analysis and evaluation of mechatronic systems
- Design and simulation methodologies
- Evolutionary algorithms in mechatronics
- Aggregation techniques in multi-criteria decision making

1.3.1 Analysis and Evaluation of Mechatronic Systems

Various mechatronic design criteria have been addressed separately in the literature on mechatronics. Dynamic interaction between different subsystems of a mechatronic system has been argued as the most important motivation for looking for new methods of analysis and evaluation of mechatronic systems. Considerable amount of work exists, which proposes important performance specifications and design criteria for mechatronic systems.

De Silva (2003) argued that dynamic interaction between components of a mechatronic system is an undesirable in the context of a sequential design, causing different subsystems to operate not in their optimal conditions compared with the conditions of sequential design of the subsystems. He proposed to define separate performance indices for different subsystems involved in a mechatronic system, which would be satisfied in a sequential design. An objective function termed "Mechatronic Design Quotient (MDQ)" was expressed using these design indices for the subsystems. The objective of the mechatronic design is then to maximize the MDQ, which will make the index after the subsystem interconnection as close as possible to their separate optimal values in the sequential design.

Moscrop *et al.* (2000) indicated that an important issue in mechatronic design was component matching. They worked on the motor-load matching aspect in three common machine tool and drive configurations. They provided a thorough mathematical analysis of the factors that interact to introduce stability problems in the presence of a high motor-load mismatch. Their results showed that mismatch between components could cause serious problems like instability in the system.

A mechatronic machine usually incorporates a degree of intelligence. Designers usually try to design autonomous and user-friendly machines. There exists work on the evaluation of the intelligence of a machine. Park *et al.* (2001) proposed a method to measure a machine intelligent quotient (MIQ) of human-machine cooperative systems. The machine intelligence was divided into two components: control intelligence and interface intelligence. A model for human-machine cooperation was presented. The system was modeled by an intelligent task graph (ITG) with the aim to explicitly separate the machine contribution from the human contribution in a human-machine cooperative system. They proposed a new human-oriented method to estimate the MIQ as a tool to help the engineers in assessing intelligent machines that are designed to possess human-like qualities with a high degree of autonomy.

Bien *et al.* (2002) proposed another method to measure machine intelligent quotient (MIQ). They carried out a literature survey on intelligent machines and extracted four common

constructs for such machines: autonomy, man-machine interaction, controllability for complicated dynamics, and bio-inspired behavior. They proposed several variables for each of these key features of an intelligent machine. They suggested two three-dimensional models to represent the intelligence of a machine, one for a large-scaled machine and the other for an intelligent robot. In order to quantify the intelligence of a machine, they adopted two fuzzy integrals; specifically, Sugeno and Choquet fuzzy integrals.

There is considerable work in functionality assessment of mechatronic systems. In this category, reliability assessment has the largest number of publications. Many authors have proposed to consider reliability as a design criterion for development of a mechatronic product (De Silva, 2003). They have argued that an optimum mechatronic design methodology is expected to provide better reliability compared with traditional electro-mechanical design methodologies.

The methods of fault tree analysis (FTA), and failure modes and effect analysis (FMEA) are common tools for reliability analysis. The main strength of these methods lies in that they have been commonly used by system engineers and have been standardized to some extent. They can be classified as top-bottom methods which emphasize on low level faults. Fault tree analysis is the process of reviewing and analytically examining a system to emphasize lower level fault occurrences, which directly or indirectly contribute to major faults or undesirable events. First, the top events or faults are specified for analysis and then an event or a series of events that directly contribute to the top-level events are identified. This process must continue until the lowest level is defined or the basic level is reached. An FTA is generally carried out for every feared critical event. The causes of the feared event and their causality connections are specified with the aim to find root events. The occurrence probability of the feared event is calculated from the occurrence probability of root events. The system analyzer needs to specify a probability density function (PDF) for each root event through experimental data or mathematical models used in this context. This analysis is based on a structural representation of a system. Nevertheless, it is only suitable if the system architecture in unchanged (Guerin *et al.*, 2002). Cassanely *et al.* (2003) have considered the reliability of automotive electronics, and have applied the method on the control system of a diesel engine. Working with FTA/FMEA analysis, they have concluded that this kind of analysis hardly helps a hardware specialist in connecting physical causes to the final effects.

In a typical mechatronic system, the actuator components rely on the measurements provided by the sensory components and the computed command signals provided by the controller

components. Guerin *et al.* (2002) have argued that failure of a mechatronic system may occur in any of these components or a particular failure mode may be caused by undesirable interactions between the components. In other words, dynamic interaction between components of a mixed system is a key issue in selecting a reliability assessment method.

Demmou *et al.* (2004) have classified FTA methods as static analysis; hence, not sufficient to deal with complex and hybrid systems in the category of mechatronics. Interaction between subsystems and the reconfiguration of a complex system cannot be properly considered by these conventional approaches.

In dealing with the dynamic characteristics of a complex system, two methods have been addressed in literature: the Markov approach and Petri nets (PN). In the Markov approach, the probabilities of different events are considered time-variant and then differential equations governing these reliability functions are derived. The main drawback of this method is that the number of differential equations increases exponentially with the number of the states in the system.

Guerin *et al.* (2002) have proposed that beyond ten components, only the PN associated with a Monte Carlo simulation is usable. PN allows modeling the functional behavior of the parts and also modeling the interactions between them. Unexpected behavior can arise due to interaction between parts. The PN model can describe the complete behavior of a system. They have used PN analysis for reliability analysis of an antilock braking system (ABS).

Dutuit *et al.* (1997) have investigated the dependability assessment by using stochastic Petri nets (SPN). They have considered SPN to deal with reliability assessment of systems where the physical behavior of the system cannot be made independent from the probabilistic behavior, as in the case of dynamic process systems. They have analyzed two case studies and have compared their results with published results. The matching was found to be satisfactory. They have then proposed SPN to model and evaluate process control systems and argued that this technique was more suitable than modeling methods like the Markovian approach.

Moncelet *et al.* (1998) looked into another branch of PNs, called colored Petri nets. This is a version of Petri nets which are applicable for systems where both continuous and discrete events need to be modeled, which is accomplished by using explicit algebraic equations in the modeling of the continuous part of the system. This provides the ability to model all the parts of the system in a single modeling language. They have argued that for a quantitative reliability assessment it is necessary to take time into account; e.g., the delay of a transition, delay related to repair and fault process. These delay times are usually modeled by random variables with

exponential distribution functions. By applying the method on a hybrid system, they concluded that colored Petri net was well-suited to describe the behavior of a mechatronic system. The method could be used for both qualitative and quantitative analysis.

Demmou *et al.* (2004) developed a method to derive critical scenarios from the Petri net model of a mechatronic system. This technique contributes to the safety evaluation and selection of system architecture in the design stage. A model combined of Petri nets and differential equations was used in order to address both continuous and discrete parts of a system. The method is based on qualitative analysis of the Petri net model of a system. They used linear logic to get a new representation of the Petri net model from causality point of view. It searches for the causality between two partial markings (a feared one and a nominal one) by carrying out forward reasoning and backward reasoning. The concept of context enrichment has been introduced in order to avoid the exploration of all the normal states and to focus on the feared states. It is accomplished by adding tokens to some places that can have an impact on the critical scenario.

Aging tokens or tokens with memory is another concept added to Petri net modeling by Volovoi (2004). The aim of this concept is to facilitate Petri net modeling. The resulting framework provides for flexible and transparent graphical modeling with excellent representational power that is particularly suitable for system reliability modeling with non-exponentially distributed firing times. In cases where the damage accumulation is changing; e.g., in multiphase missions or due to load sharing, previously the change in system configuration had to be modeled by the so-called marking-dependent firing policies, because memory was associated solely with transitions. In addition to modeling difficulty, there was no accepted way to express such dependence in a simple graphical way. It had to be described using external logic, resulting in losing clarity. In addition, if an event existed in the behavior of the system that changed the properties of a component, it would not be a proper way to model it by the previously used Petri net modeling methods. The concept of aging tokens was introduced by Volovi to deal with such difficulties in modeling. It has significantly improved the flexibility and clarity of SPN.

1.3.2 Design and Simulation Methodologies

Integrated mechatronic design can be greatly facilitated by a software tool that supports modeling and simulation of physical systems, together with their controllers. There is considerable literature presenting bond graph (BG) simulation as a domain-free approach that

provides a core language to model subsystems from different engineering disciplines in a unified language, and hence particularly appropriate for modeling mechatronic systems.

Granda (2002) explained the steps for developing a bond graph simulation tool called CAMPG (computer aided modeling program with graphical input). It was shown that state space dynamic equations and also transfer function of a system can be derived from its bond graph model. Amerongen (2003) used another bond graph simulation tool, called "20-sim", for mechatronic design of DC servo system. It was shown using several examples how bond graph simulation software could help to find proper parameters of mechanical components and of the controller. Borutzky (2000) elaborated modeling of hydraulic components using a bond graph modeling tool called "Modelica."

Bond graph modeling can be considered a special case of object-oriented modeling (OOM), although BG was developed far earlier than the concept of OOM (Borutzky, 1999, Borutzky, 2002a). BG modeling and OOM have much in common, in particular in terms of domain independency, support of hierarchical modeling, and non-causality of sub-models. Borutzky (2002b) proposed a systematic method to determine non-normalized frequency domain sensitivities in symbolic form. In this method a bond graph model called "associated incremental bond graph" is constructed from another bond graph. In associated BG, the variables associated with the bonds are incremental changes in the power variables from their nominal values, due to incremental parameter changes. By deriving the system equations in symbolic state space form from the incremental BG, in the same way as they are derived from the original BG, the sensitivity matrix of the system can be set up in symbolic form.

There is considerable literature on bond graph modeling of complex mechatronic systems, including a nonlinear shock absorber (Mollica, 1997), a hydraulic brake system (Kuang, 1999), a rotary crane (Sagirli, 2003), a hydraulic motor/pump for dynamic braking (Lumkes, 2000), and a flow orifice for laminar and turbulent conditions (Burutzky, 2002c).

There exists some work on the development of mechatronic design methodologies. Van Brussel (1996) defined mechatronics as the science of motion control and mentioned control bandwidth and path accuracy as key features in precision machines and mechatronic designs. He argued that a concurrent approach is necessary in the design of a mechatronic product .One of the challenges in the development of a concurrent methodology is the combination of distributed models like finite element (FE) models with control models. Finite element models can have a very large number of degrees of freedom which may not be suitable or necessary for control

models. He proposed to create a "mechatronic compiler," which would be able to accept high-level system specifications and run them (semi)automatically, into optimal system parameters.

Van Brussel *et al.* (2001) proposed a method to derive low order dynamic equations from a finite element model for use in control models. This method is based on component mode synthesis (CMS) techniques. After addressing the challenges of optimal mechatronic design, they proposed a simultaneous engineering framework for designing optimal mechatronic systems. A three axis machine tool was used as the case study in this work. Geometric configuration of the machine was optimized by a genetic algorithm in the conceptual stage of the design. A method based on component mode synthesis was used in order to derive a low order control model from the finite element model of the mechanical and drive structures of the mechatronic system. In the last stage, motion controllers were derived to be robust against configuration changes.

Zhang *et al.* (1999) proposed an integrated mechatronic design methodology for the mechanical structure and the control algorithm in a programmable closed-loop mechanism system. From the mechanical structure viewpoint, they dealt with the mass distribution in the linkages of the mechanism. The overall idea was to design the mechanical structure so as to facilitate controller design. To realize this objective, they tried to design the mechanical part so that it resulted in a simple dynamic model, and hence providing a more predictable dynamic response. The mass-redistribution was determined based on the principle of balancing both shaking force and shaking moment, because of the harmful effects of them to the machine environment.

Li *et al.* (2001) extended the pervious work in order to develop a concurrent mechatronic design method. They first expressed a mathematical model to represent a mechatronic design. In this expression, the design problem was considered as a mapping from the "behavior space" to the "parameter apace." Each of these spaces is divided into two parts: real-time and non-real-time. By this representation of a concurrent mechatronic problem, they verified the need for a concurrent design approach in developing mechatronic systems. They introduced the idea of design for control (DFC) which was a concurrent framework facilitating the controller design. Then, a concurrent design was carried out with the aim to achieve simple dynamic equations for which the design and implementation of a controller would be easier. They used as a goal in mechatronic design, the reduction of the shaking force/ moment of the actuators.

Pil *et al.* (1996) studied the integrated design of mechatronic systems from a quite different viewpoint. They established a concurrent design method using a recursive experimental

optimization method. Control gains were optimally tuned in a prototype of the machine, and its mechanical structure was physically modified to improve the control performance. The entire process of controller tuning, performance evaluation, and structural modification was repeated until the desired performance goals were achieved. In each iteration, incremental design changes were determined based on a sensitivity Jacobian relating structural design changes to performance improvement. The actual data of the design changes were used to update the sensitivity Jacobian. A rapid prototyping technique was used to evaluate the performance in each design iteration, and estimate the sensitivity Jacobian. To verify the design approach, they applied it to the design of a robot positioning system. Through a recursive procedure, an optimal combination of the arm structure (with regard to stiffness and damping) and proportional-derivative (PD) control gains was achieved.

De Silva (2003) proposed that optimum and concurrent mechatronic design can be treated as an optimization problem if proper evaluation methodology is established. A mechatronic design, in view of its unified and synergistic treatment of components, should be better than a traditional design where electrical design and mechanical design are carried out separately. He looked at concurrent design from a different viewpoint: trying to reduce the undesirable effects associated with sequential design. Dynamic interaction between components was considered the most effective source of deviation of the result of a sequential design form an optimal concurrent design. In traditional uncoupled design of a mixed system, the electrical subsystem is designed by considering the mechanical part as a constant load, and the mechanical subsystem is designed by looking at electrical part as a fixed energy source. Once they are interconnected, neither electrical nor mechanical objectives would be satisfied in the expected level, due to interaction between them. Once the optimal sequential (uncoupled) designs of the mechanical and electrical subsystems are identified, the corresponding design indices can be identified. By formulating a proper "mechatronic design quotient (MDQ)" in terms of these optimal design indices of the uncoupled subsystem designs, he presented the objective of optimal concurrent design as maximization of MDQ, which brings the performance satisfaction of the subsystems when they are interconnected as close as possible simultaneously to the reference case of optimal uncoupled subsystems.

Coelingh *et al.* (2002) formulated an assessment method for conceptual design of electromechanical motion systems. The method was based on a classification of standard transfer functions, plant models, and closed loop systems. The aim of the method was to provide

an integrated way for the design of the path generator, control system, and electromechanical plant with appropriate sensor locations.

Moulianitis *et al.* (2004) proposed a method for conceptual mechatronic design based on a model they expressed for concept evaluation. During conceptual design, candidate solutions are created and evaluated. They proposed an evaluation index to be calculated for all candidate conceptual designs to help the designer in making decisions. Their evaluation index included the three elements: intelligence, flexibility, and complexity of a mechatronic product. The evaluation was carried out by calculating the score corresponding to each of these criteria. Some guidelines and preliminary formulas were given in this work for evaluating each of these criteria in the early stages of the design. Weight factors were applied to highlight the importance of each criterion. The overall evaluation was formulated based on the t-norm and averaging operators. The variety of t-norms and averaging operators were used to show the weaknesses and strengths of them. They considered Werner 'fuzzy and' in the evaluation process and argued that it had a more optimistic sense than the minimum operator combined with a more pessimistic sense of the mean operator. The underlined case study in this work was the conceptual design of robot grippers for handling fabrics.

Avigad *et al.* (2003) presented a preliminary work on development of a tool for mechatronic design, with emphasis on the simultaneous mechanical and control designs. A mathematical model was presented for mechatronic design. They expressed the framework for a genetic-algorithm-based design methodology that is particularly developed for exploring possible design alternatives in early stages, with an emphasis on interdisciplinary design. As part of this design approach, a definition for a discipline related idea in interdisciplinary search space, called "sub-concept" (S-C) was introduced and implemented for mechatronics. An S-C was defined as an abstract description of part of a solution. A mixture of S-Cs from one or more disciplines could also be defined as an S-C. By defining and coding these S-Cs, they created an individual structure called Compound-Individual (C-I).

1.3.3 Evolutionary Algorithms in Mechatronics

Evolutionary algorithms are increasingly used in various engineering problems. In particular, genetic algorithms are useful in optimization problems when it is difficult to establish an explicit analytical relationship between design parameters and objectives of optimization. Then, the particular physical system is in fact a "black box," and may not have a precisely defined solution method, or if it does, the solution is infeasible due to complexity and/or lack of the

needed computing capabilities. Genetic algorithms are inspired by the evolutionary process of species in nature. Many potential (trial) solutions are generated randomly and encoded on a simple chromosome-like data structure. These random solutions are analyzed to see how satisfactory the result of each trial solutions is. Reproduction operations are applied so as to preserve critical information and in such a way that better solutions have a greater chance to participate in the reproduction operations. Accordingly, in this approach, it is not needed to "formulate" how to solve the problem, and only necessary to evaluate how well each trial solution compares with the desired behavior/response.

Chocron *et al.* (1999) proposed a design method for a class of mechatronic systems, specifically locomotion systems, using genetic algorithms. A genetic algorithm was used to adopt both mechanical structure and the controller of a walking robot for various locomotion tasks. The objective was to achieve movement over complex surfaces. To perform adaptation of the mechanical structure, a modular robotic system (MRS) was used, which was formed by assembling elementary mechanical modules. In order to simultaneously design the mechanical structure and the controller, hybrid encoding was used. To evaluate the trial solutions, a dynamic simulation was integrated within the genetic algorithm, which computed and exploited all criteria in real time.

Genetic programming (GP) is another branch of evolutionary algorithms. The main difference between GP and conventional genetic algorithms is in the representation of trial solutions. In GP, each trial solution is represented by a geometrical tree-like structure. The branches of this tree-like representation are construction functions to form a trail solution from an initial embryo model. This is a powerful search tool, which was initially used to explore mathematical equations and automatically generate computer programs (Koza, 1999).

Koza *et al.* (2000) used the power of genetic programming for the synthesis of topology and sizing of electrical circuits. He established a mapping between the tree-like representation of solutions in GP and labeled graphs germane to electrical circuits. The mapping was established by defining several *component construction functions* which were applied to an *embryo* model to generate a variety of possible circuits. By exploring the topology of the circuit using this mapping, both topology and sizing of an electrical circuit could be designed automatically and optimally. The result was an automated tool for the design of electrical circuits using high level statements of the desired behavior of the circuit. He applied this tool in the design of eight commonly needed electrical circuits, including a low-pass filter, a high-pass filter, a band-pass filter, a tri-state frequency discriminator circuit, a frequency measuring circuit, a 60 dB

amplifier, a computational circuit for the square root function, and a time-optimal robotic controller circuit.

Seo *et al.* (2003) used the idea of Koza's work and integrated GP with bond graph modeling to create a unified tool for automated design of mechatronic systems. Bond graph modeling was used in view of three reasons. First, it is a domain-independent modeling tool; hence suitable for the modeling of mixed mechatronic systems. Second, its graphical representation allows free and open-ended composition of new models. Third, it is efficient for the classification and analysis of models, allowing rapid determination of various types of acceptability or feasibility of candidate designs. They introduced some construction functions and terminals to establish a mapping between BG models and a GP tree-like representation of a trial solution, and benefited from the high search power of GP to explore a wide range of topologies in the design process. By this method, both topology and sizing of a mechatronic product could be optimized. The suggested method was applied in three design examples: first, a domain independent eigenvalue placement; second, an analog electrical filter to achieve specified performance over a given frequency range, and third the redesign of a printer drive system to obtain the desirable steady-state position of a rotational load. This work considered only the structural part (mechanical, electrical, hydraulic, etc.) of the mechatronic systems and missed the controller part which is always present in a mechatronic system. Taking into account the interactions between the structural part (non-real-time parameters) and the controller part (real-time parameters) is the main challenge, and is an important objective of mechatronic design, which is addressed through a concurrent design methodology. On the other hand, the developed construction functions could only generate linear components; hence the developed tool was only applicable to linear problems.

Wang *et al.* (2005) extended the integration of bond graph modeling and genetic programming to acquire knowledge through evolutionary computation as applied to conceptual design of mechatronic systems. In contrast with the previous work, the focus of this work was mainly on the controller part of a mechatronic system. "Controller design in the physical domain" was used as a means to unify the design of control system with the design of mechanical systems, in the bond graph modeling environment. In this method, a controller is represented by a physical equivalence combination of bond graph elements C, I, and R. Various combinations of these elements can be generated to represent different control schemes such as proportional (P), proportional plus derivative (PD), proportional plus integral (PI), and proportional-integral-derivative (PID), or lead and lag compensators. It was shown that by

structuring the design primitives into modular building blocks inductively and deductively, knowledge interaction was significantly facilitated. Results from this work showed that by applying genetic programming to evolve bond-graph control structures, there was the potential to discover good control strategies that might not be obtained through conventional methods.

Optimization of multimodal functions is the subject of a branch of evolutionary algorithms called niching genetic algorithms. This arises when the function to be optimized has several local peaks and one global peak, and the user is interested to know all optima in addition to the best solution (global optimum). In dealing with multimodal functions, a simple GA cannot maintain controlled competition among individuals corresponding to different peaks; hence, the population converges to one alternative or another. In the case of unequal peaks, a simple GA converges to the best peak.

There exist different strategies to implement niching genetic algorithms. Himeno *et al.* (2003) proposed a method called "Dispersing Deterministic Crowding" (DDC). It encourages dispersion of individuals and creation of species within the population in order to increase the discovery of local optima, as well as the global optimum. Also, they described other methods of niching GA and compared with DDC, including deterministic crowding (DC), sharing, GA with tabu search, restricted tournament selection (RTS), and immune algorithms. In a simple GA, once the reproduction operation is carried out, the created children are used to replace the parents in the population. In deterministic crowding, individuals from the population are randomly paired off and recombined. Unlike the selection process of a simple genetic algorithm, here all individuals have equal chance to be selected, regardless of their fitness. After the reproduction operation is carried out, the four individuals, parents and children are regrouped into pairs with the highest mutual similarity. The children replace the parents only when the fitness of the children is higher than that of the parents. In DDC, the locations within the respective peak regions of both the new individuals and the parents are checked and the individuals with a higher position relative to its peak are kept for the next generation. Comparing with the results of other methods, the authors of this paper claimed that their method was superior, because it discovered most of the solutions, including local optima.

Cho *et al.* (2001) used niching GA for the design of an induction motor for an electric vehicle. Niching GA contributed to the design of induction motors by allowing integration of different criteria in the optimization process. Due to computation time restrictions, it was difficult to analyze all the criteria in the course of optimization. It was proposed to consider only the most important criteria in applying niching GA. Once elite solutions with respect to this essential

criterion were found, they were post-processed using other criteria to find the best solution. They suggested using the niching algorithm with the implementation technique called restricted competition selection (RCS). This method is very similar to what happens in nature. In natural ecosystems, a niche can be viewed as an organism's task, which allows species to survive in their environment. The subdivision of the environment on the basis of an organism's role reduces the interspecies competition for environmental resources, and this reduction in competition helps formation of a stable subpopulation around different niches in the environment. In the RCS method, competition is restricted among dissimilar individuals during selection to reach a stable subpopulation around each local optimum. From competition among similar individuals, where their difference in the search space is smaller than the niche radius, the loser's fitness is changed to zero. Consequently, only the best individual per niche is maintained, and an elite set is introduced to preserve the local optima obtained during optimization. In particular, this feature of RCS is well-suited for engineering design applications. In an engineering design, only the best individual per each niche is needed, because individuals in the same niche have a similar configuration.

Kim *et al.* (2002) proposed a neo-niching method using the RCS method combined with the pattern search method (PSM), aiming to reduce the calculation time. Through the application of this new method to a numerical example and the optimal design of an interior permanent magnet synchronous motor, they were able to achieve increases in the speed of searching for peaks.

For improving the computational efficiency of genetic algorithms, several workers have proposed to involve *memory* for design variables. A genetic algorithm with memory tries to reduce the number of fitness function evaluations required by the genetic algorithm (Kogiso *et al.*, 1994; Gantovnik *et al.*, 2003). By involving memory in GA, information from previously analyzed design points is utilized in later searches with the aim to avoid repeating analysis of previously encountered designs. Kogsino *et al.* (1994) introduced the concept of memory only for discrete variables whereas Gantovnik *et al.* (2003) proposed an algorithm for GA with memory that could work with discrete and continuous variables simultaneously.

1.3.4 Aggregation Techniques in Multi-Criteria Decision Making

One of the contributions of the work presented in this thesis is to present a system-based model for optimal mechatronic design. There are a variety of interactive criteria involved in mechatronic design. System-based viewpoint toward the design strives to improve an overall degree of satisfaction of the criteria of system design, through an intuitive aggregation of

different and sometimes opposite design objectives. Aggregation of criteria in a multi-criteria problem and possible interactions which may exist have been addressed in literature. In this section, the literature on multi-criteria aggregation using fuzzy logic is reviewed.

Grabisch (1996) illustrated the application of fuzzy integrals in multi-criteria decision making. He presented a history of fuzzy integrals, which were first introduced in 1974 by Sugeno. In the discrete case it is merely a kind of disordered mean. The distinguishing feature of fuzzy integrals is that it is able to represent interaction between criteria. Prior to this tool, there was no well-established method to deal with interacting criteria; hence, then the problem was merely avoided by constructing independent criteria. Grabisch represented the requirements which an aggregation operator should possess, both in terms of mathematical properties and behavioral properties. Then he showed that two commonly used fuzzy integrals, Choquet and Sugeno, had the required properties. The drawback of fuzzy integrals was recognized to be its complexity due to exponential growth of number of coefficients that must be specified with the number of criteria. A method for identification of these coefficients, based on learning data was proposed in this review paper.

Marichal (2000) represented the axiom that supports the Choquet integral, and also showed an intuitive interpretation of its operation. Different possible interactions which may exist in a multi-criteria problem were addressed and illustrated by using simple examples.

Marichal (2001) represented another fuzzy integral capable of addressing interactions between criteria, called the Sugeno integral. It is used particularly when available information is of qualitative nature. The application of the Sugeno integral as an aggregation function was justified by an axiomatic approach. It was shown that mutual preferential dependence of criteria reduced the Sugeno integral to a dictatorial aggregation.

Bien *et al.* (2002) presented a method for the measurement of machine intelligence quotient (MIQ), which was based on aggregation of several criteria using fuzzy integrals. They argued that Sugeno fuzzy integral rendered a kind of weighted median, and thus produced a very conservative value for calculation of the machine intelligence quotient. They proposed using the Sugeno integral for subjective evaluation of a machine. The Choquet integral, on the other hand was considered as a special form of weighted average, offering the global degree of intelligence of the system. They proposed to use the Choquet integral for an objective evaluation of the intelligence of a machine.

1.4 Contributions and Organization of the Thesis

The main contributions of this thesis are outlined now.

1. System-based mechatronic design is presented as a new viewpoint and approach to achieve truly optimal mechatronic designs. As discussed in the literature, an optimum mechatronic design must utilize a concurrent and integrated approach. In this thesis, system-based thinking is included as another requirement to achieve optimal mechatronic design. There are a variety of criteria involved in a mechatronic design problem. These criteria commonly have interactions with each other, and they may even exhibit negative correlation with each other. To achieve optimal design, maximizing a general sense of satisfaction of the product has to be considered as the design objective. The Mechatronic design quotient (MDQ) is revisited in this thesis as a multi-criteria evaluation index representing the score of general satisfaction of a mechatronic design objective. It is calculated using soft computing methods, to account for interactions between criteria and reflect the expert's experience on mixed systems. The MDQ supports the system-based mechatronic design and serves as a fitness evaluation index for design candidates.

2. A new method is suggested for optimum conceptual design of a mechatronic product. This method is constructed using the MDQ as a competition index between possible conceptual choices for a particular product. The MDQ contributes in the design of a mechatronic product by guiding the designer and helping in making decisions in early design stage. The developed methodology provides an integrated, concurrent, and system-based approach, which contributes in making best decisions with respect to the criteria reflected in the MDQ.

3. The most challenging issue in considering multiple criteria in the optimization process is the computation cost associated with the evaluation of various criteria in the course of the optimization process. A two-phase MDQ optimization approach is suggested to resolve this difficulty. Niching genetic algorithm is adopted for the first phase, with the aim to find an optimal representative for different possible configurations, with respect to some essential MDQ attributes with veto effect on the final evaluation of design candidates. In the second optimization phase, a full and comprehensive MDQ evaluation is carried out to find the best solution among representatives of each configuration.

4. A new approach is developed for reliability assessment of repairable mechatronic systems. Two main issues in reliability assessment of complex machines are addressed in this approach; namely, dynamic interaction between subsystems and severity of failure modes. The real operation of a large number of machines of a particular type is mimicked using generalized stochastic Petri nets (GSPN). A severity index for each failure mode is calculated using the Choquet integral for aggregation of the partial scores to different criteria involved in the severity of a failure mode. To quantify the "feeling" of machine reliability in customers, an equivalent time-varying reliability specification is proposed which reflects dynamic interactions between subsystems of a machine and also the severity of failure modes. This reliability assessment approach contributes to the developed design methodology by providing information for making appropriate decisions about the overall architecture of the machine and also on issues such as component redundancy, control hierarchy, repair policies, and so on.

5. An original work is carried out in the integration of genetic programming and bond graph modeling for nonlinear mechatronic problems. It represents the first effort to extend the integration of these two powerful tools for nonlinear problems. An original application of this unified tool to automated system identification of a complex mechatronic system is presented, where both topology and sizing of the model are optimized.

The subsequent chapters present the research that leads to the mentioned contributions. The organization of these chapters is as follows.

Chapter 2 describes the concept of the mechatronic design quotient (MDQ) and explains its calculation and application in system-based mechatronic design. Multi-criteria decision making is explained in this chapter. In particular, possible interactions which may exist between multiple criteria are explained. The Choquet fuzzy integral is presented as an appropriate aggregation tool for multi-criteria decision making. The framework of a systematic mechatronic design methodology is outlined in this chapter. An industrial fish-cutting machine is used as the test bed for the present research and development activity. The operation of the machine is described in this chapter.

Chapter 3 presents the methodology which is proposed for conceptual mechatronic design. The underlying steps for using the MDQ in the conceptual design stage are presented. Some guidelines on evaluation of mechatronic design criteria in the early design stage are given in this chapter.

Chapter 4 presents the underlying process for the detailed design stage. The difficulties of MDQ optimization are discussed. A new two-phase optimization approach is presented. The niching genetic algorithm and the approach of its implementation are described in this chapter.

Chapter 5 presents the development of a bond graph simulation tool which supports use of genetic programming as an optimization tool. A new matrix-form formulation is presented which facilitates the creation of a bond graph tool.

Chapter 6 presents work on the integration of bond graph modeling and genetic programming for nonlinear mechatronic problems. The construction functions which provide a mapping between a GP tree-like representation of trial solutions and bond graph models are presented. The automatic modeling of a vibratory system and the electro-hydraulic manipulator of a fish cutting machine are illustrated.

Chapter 7 provides the framework for the developed reliability assessment method for repairable mechatronic systems. The construction of a Petri net model for reliability analysis is illustrated. Calculation of the severity index for failure modes is presented. Petri net modeling and reliability assessment of fish cutting machine are described in detail. Some typical questions on reliability analysis are investigated in this chapter, such as the effect of the controller, component redundancy, and fault detection and diagnosis, on reliability.

Chapter 8 concludes the thesis by summarizing the overall research which has been carried out and the significant contributions thereof. Directions for future research in the thesis area are indicated as well.

Appendix A describes the Petri net method for simulation of the dynamic architecture of a complex machine and evaluation of its functional behavior.

Chapter 2

Mechatronic Design Quotient (MDQ)

A mechatronic product is a mixed system which ideally needs an integrated and concurrent design approach (De Silva, 2003), with the aim to achieve better component matching, increase efficiency, reduce cost, facilitate system integration, create proper cooperation with other systems, and increase reliability (Behbahani and de Silva, 2005a, Behbahani and de Silva, 2005d, Behbahani and de Silva, 2006b). In other words, a system-based and multi-objective thinking method is ideally needed for a mechatronic designer due to complexity of interactions between subsystems of a mechatronic product. This ideal thinking attitude will imply that the best overall system performance with regard to a variety of criteria, which are important in reality, cannot be achieved unless a concurrent and integrated design method is employed.

In conventional design methodologies, which can be classified as sequential, different subsystems are designed separately and optimized with regard to the particular criterion which is important for that subsystem. The off-the-shelf components are selected by considering only some of the capabilities, while some other performance specifications may not be considered in the selection process. For example, when purchasing an electrical motor, it is selected primarily based on the required power and price limitations, while there are many other characteristics in an electrical motor that affect the performance and limitations of the controller, such as torque constant, back e.m.f. constant, mechanical inertia, resistance, inductance, and damping (De Silva, 1989; 2005). These specifications, which may not be considered in the motor selection, can make a significant impact on the controller design for the intended task, apart from the required power. Once all subsystems of the system are designed and built, they are interconnected together and a controller is designed to carry out the intended tasks of the system. Due to interaction of subsystems, the performance of the system can degrade with respect to some attributes. In other words, although each individual subsystem is designed optimally, they may be far from their expected optimum operation when they are interconnected, due to the interactions between them. The designer would then try to overcome these problems and achieve better performance by designing a sophisticated controller, even though perfect performance might not be achieved due to hardware limitations. For a mixed mechatronic system, it is unrealistically optimistic to expect the final product of such a design process to be optimal.

A concurrent and integrated design approach will not be easy, however, because of the diversity of design objectives, complexity of possible interactions between components, need of knowledge in a variety of fields, and the lack of a formal and systematic design approach. Due to these difficulties, most design engineers still follow traditional sequential design methods which typically do not lead to optimal designs.

This chapter presents the concept on mechatronic design evaluation called mechatronic design quotient (MDQ), which was first proposed by De Silva (2003). MDQ is a multi-criteria design evaluation index, reflecting the degree of overall satisfaction of the design criteria for a mechatronic system.

Since most of existing mixed systems have been designed through a sequential approach, they may not be optimal. There is potential for improvement through concurrent design. MDQ can be used for existing mixed systems to help the designer evaluate how well the product has been designed and also find out components or subsystems which can be improved. It can also be used for commercial purposes to rank different products from a technical point of view.

In this chapter, the framework of a new systematic design approach will be elaborated for concurrent and integrated design of a mechatronic system by using the concept of mechatronic design quotient. The overall idea of this design methodology is that an ideal mechatronic design can be treated as an optimization problem if there is an intuitive, multi-objective, and well established design evaluation method reflecting all the objectives of concurrent and integrated design. The concept of MDQ is used to represent a mechatronic design evaluation scheme. The developed design methodology is a multi-stage process and MDQ is the essential base of all stages. The underlying process of the developed approach can be treated as concurrent, integrated, and system-based, which results in an optimal design with respect to the criteria reflected in MDQ.

2.1 Model of Mechatronic Design

An engineering design problem may be modeled as a mapping from a required behavior space to a parameter space without violating a specified set of constraints. Alternatively, this may be considered as progressing from a specification phase to a realization phase. For a mechatronic design, Li *et al.* (2001) proposed to divide the requirement space into two subspaces (Figure 2.1) which represent:

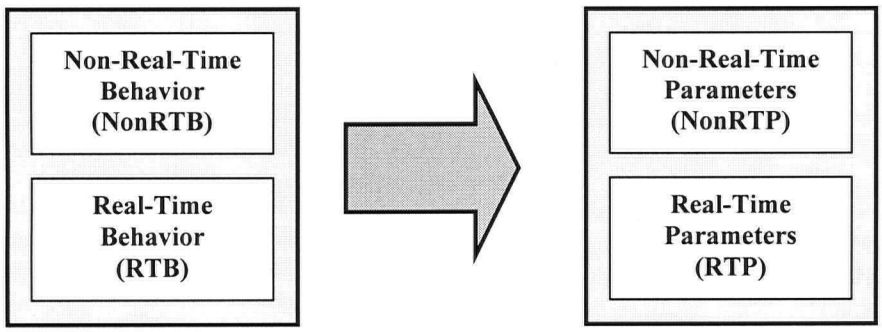


Figure 2.1: Mapping of a mechatronic design.

- Real-time behaviors (RTB)
- Non-real-time behaviors (NonRTB)

Following this division, the system parameters in the structural space can also be divided into two subspaces consisting of:

- Real-time and controllable parameters (RTP)
- Non-real-time and uncontrollable parameters (NonRTP).

Figure 2.2 shows some examples of these concepts as applicable to a robot. Here "real-time" denotes parameters and specifications which may change with time after the machine is built;

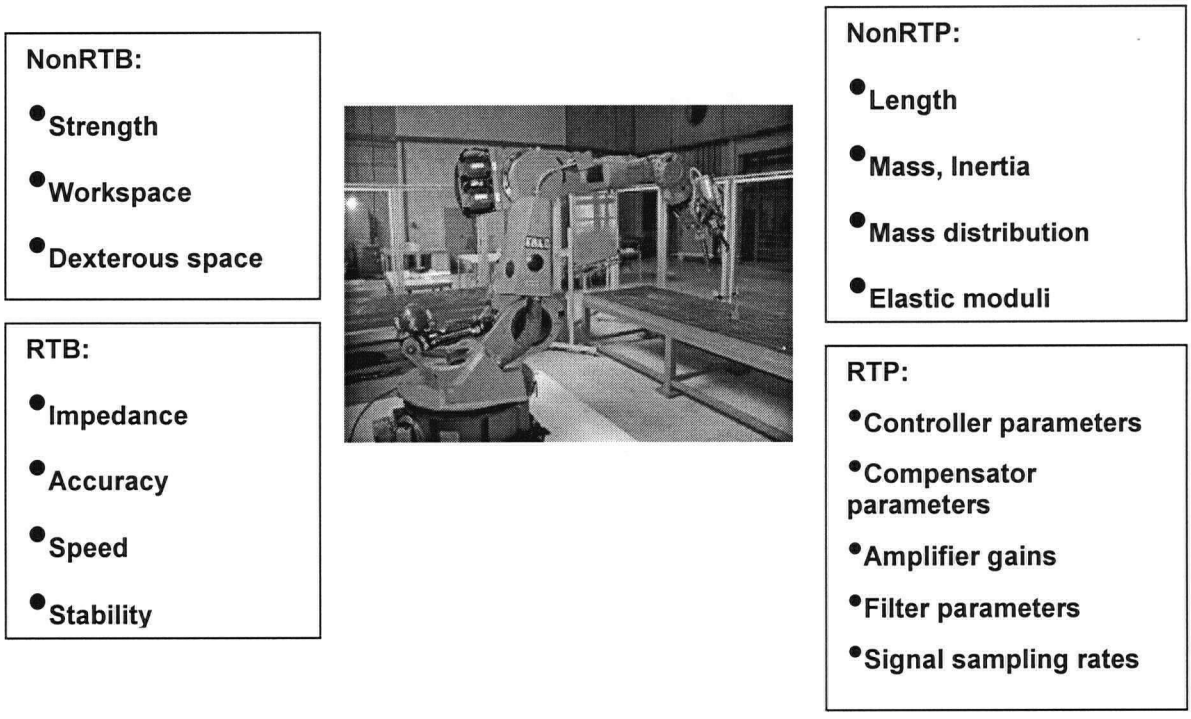


Figure 2.2: Mechatronic design concepts for an example of a robot.

for example, controller gains, time constants, accuracy, and speed. In contrast, "non-real-time" refers to parameters and specifications which may not change with time after the machine is built; for example, structural material, weight, and workspace.

In a traditional sequential design process (termed electromechanical design), NonRTP are designed first based on the NonRTB specifications (Figure 2.3). This process itself is performed in two stages.

1-The mechanical structure of the machine is designed (e.g., configuration of the machine, sizing of the structure, structural material) according to some NonRTB specifications (e.g., allowable stress, allowable workspace).

2-Electrical components are added to the machine (e.g., actuators, sensors) based on the NonRTB requirements (e.g., maximum required acceleration, minimum acceptable accuracy).

Subsequently, RTP (e.g., control algorithm, controller gains, poles, zeros, signal conditioning algorithm) are determined based on RTB specifications (e.g., desired path, speed, accuracy, stability) to control the already-established mechanical structure.

In such a traditional design methodology, it would be costly to vary a parameter in the mechanical structure or change an electrical component after the machine is built. Therefore all parameters related to mechanical structure and electrical hardware are hardly varied and can be treated as time invariant (NonRTP). In contrast, the parameters in the driving and control

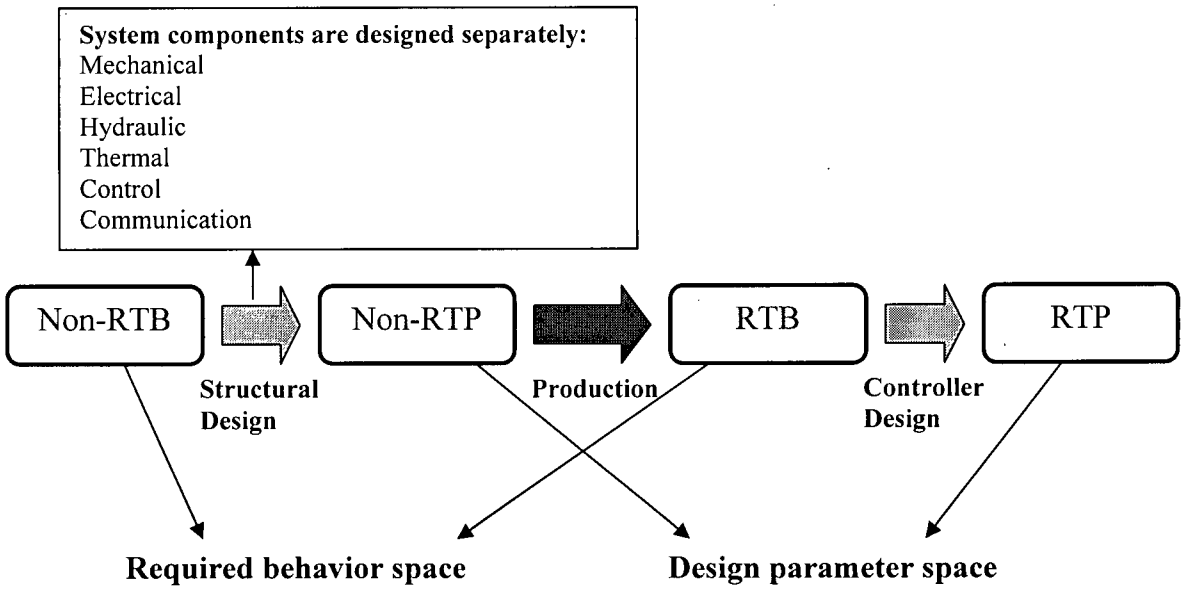


Figure 2.3: Model of traditional sequential design (electro-mechanical design)

structure are by and large changeable and can be adjusted after the system is built (RTP), which can be incorporated in the programmability of the electromechanical system. Recently, evolution of control engineering and computer science resulted in the creation of a school of thought that considered the design of mechanical structure and electrical hardware to be no longer the main design focus in some electromechanical systems, and that the inadequacies of the system mechanics could be compensated for by sophisticated control schemes and software. This thinking can be self defeatist because a perfect control action may be hardly achieved due to hardware limitations and dynamic interactions, regardless of the effort devoted to the design of the control structure. This does not mean, however, that the performance of a machine cannot be improved by better control. It is the "adequacy" and the "optimality" that are in question.

In the mechatronic design methodology developed in the present work (Figure 2.4), it is believed that controllability and programmability of RTP should be viewed as an opportunity to further improve the design after the machine is built, unlike in the traditional electromechanical design where programmability of RTP is considered an excuse to postpone their design after the machine is built. In particular, in the present approach, RTP and NonRTP are designed concurrently considering both RTB and NonRTB. After production of the machine, NonRTP may not come out to be exactly as expected during the design process. This results in deviations of RTB and NonRTB from their desired and expected conditions. To compensate for these deviations, system identification is then performed to find out the actual NonRTP. The programmability of RTP is then exploited to compensate for the deviation of the behavior of the system from desired behavior, which intuitively should not be very deep.

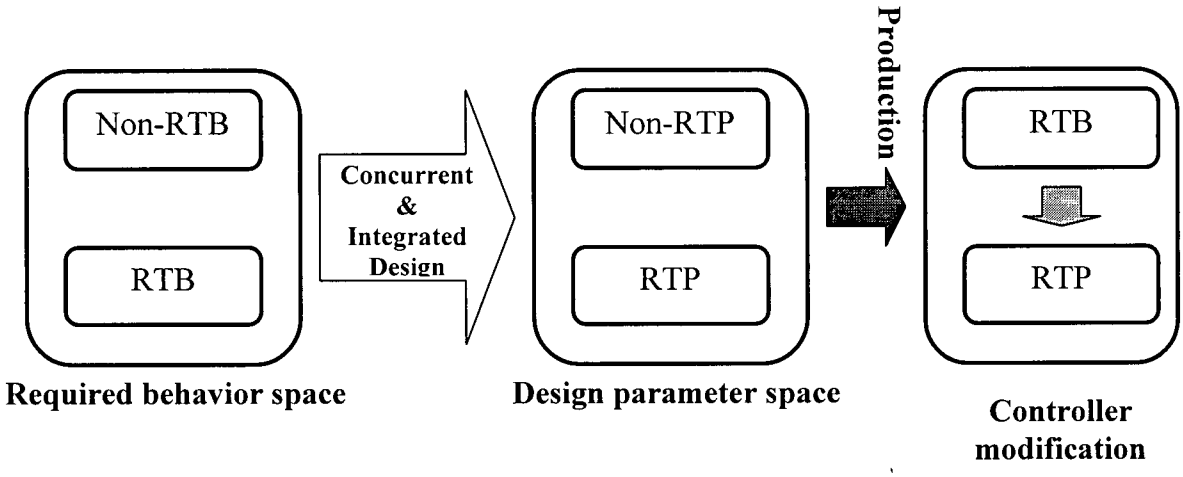


Figure 2.4: Model of the developed mechatronic design methodology.

An ideal mechatronic design should also be system-based. Each component of the system has a particular objective and has a role within the objectives of the overall system. It can have interactions as well with other components, thereby affecting their objectives. A system-based design tries to improve the product performance from different views considered simultaneously. In other words, it tries to improve the overall degree of satisfaction of the system through an intuitive aggregation of different and sometimes opposite objectives. This implies that a mechatronic design can be classified as a multi-objective problem. From a high-level viewpoint, better performance, greater reliability, and increased intelligence are expected from a mechatronic design compared to a conventional design. In particular, a mechatronic design is expected to result in better component matching, better controller performance, and greater efficiency.

MDQ is the basis of the design methodology developed in the present work. A system-based design is performed by using an intuitive aggregation of different criteria. MDQ is used as a goal function and the design problem is treated as an optimization problem. The result of this design methodology provides an optimum combination of the criteria involved in the calculation of MDQ. In other words, it provides the design which has highest global sense of satisfaction.

2.2 Mechatronic Design Quotient (MDQ)

Desired behavior of a system is represented by some design criteria. In general, the behavior of the system may be treated as a function of the system parameters (design parameters/variables):

$$Y = f(X) \tag{2.1}$$

where $X=\{x_1, x_2, \ldots, x_m\}$ is the set of design variables/parameters or decision variables/parameters and $Y=\{y_1, y_2, \ldots, y_n\}$ is the set of behavior specifications, which represent the design criteria. The design process can then be considered as a procedure of finding X_d so as to satisfy the required behavior specifications Y_d, without violating a set of constraints:

$$Y_d = f(X_d) \tag{2.2}$$

Desired behavior can take three forms:

- Maximizing a function $y_d \equiv Maximize\left[c_{\max}(X)\right]$
- Minimizing a function $y_d \equiv Minimize\left[c_{\min}(X)\right]$
- Satisfying an equality $y_d \equiv \left[c_{eq}(X) = \Omega\right]$

Two kinds of constraints may exist:

- Inequality constraints: $g_i(X) > 0$, and
- Equality constraints: $g_e(X) = 0$

If the physics of the problem provides a direct and explicit relationship between the parameter space and the behavior space, then the desired parameters can be found simply by substituting the required behavior into the equations. For example, if the required safety factor and the maximum acceptable deflection of a beam are specified, its cross-section can be calculated by using standard elasticity equations. But in most engineering problems and particularly in mechatronic design, such a straightforward relation between design variables and behavior specifications cannot be found. Then an inverse design approach should be taken, where one starts from an embryo design, modifies the design variables, and makes proper decisions to achieve the desired behavior. Due to inequality expressions in the desired behavior, the solution to this problem may not be unique. The task of the designer is then to find the design parameters that not only satisfy the desired behavior without violating the constraints but also achieve the best satisfaction—the optimal design. The idea of the proposed design methodology is that if there is an adequate and comprehensive evaluation method for a mechatronic design, then the design problem can be treated as an optimal decision-making problem. In the context of this design methodology, Mechatronic Design Quotient (MDQ) represents a multi-criteria design evaluation index which reflects the degree of satisfaction of mechatronic design criteria and indicates the meeting of constraint requirements.

Supposing that *n* design criteria and *r* constraints exist, MDQ can be expressed as:

$$MDQ(X) = H\left[s_1(X), s_2(X), \ldots, s_n(X)\right] \cdot \prod_{k=1}^{r} G\left[g_k(X)\right] \tag{2.3}$$

where *H* is an aggregation operator, $s_i(X)$ is the partial score between zero and one from the i^{th} criterion showing its degree of satisfaction, and $G[g(X)]$ is a function indicating whether a constraint has been met. It is equal to one if the constraint has been satisfied and zero otherwise. A mechatronic design problem can be treated as optimization of MDQ.

A useful definition for MDQ should span a wide range of features and needs of mechatronic systems. Following criteria are some general objectives of a mechatronic design (Figure 2.5):

1-Satisfaction of the task requirements

2-Component matching

3-Efficiency

4-Intelligence

5-Reliability

6-Controller friendliness

7-Cost

2.3 Aggregation of Criteria

An appropriate approach for criteria aggregation is presented now. For each design alternative, a partial score between zero and one is assigned to each criterion. Then the MDQ is computed by aggregating these partial scores:

$$MDQ = H(s_1, s_2, \ldots, s_n) \cdot \prod_{k=1}^{r} G\left[g_k(X)\right] \tag{2.4}$$

where H is an aggregation operator and s_i is the partial score for the i^{th} criterion.

The common aggregation technique, weighted average method, is not intuitive because it is essentially a linear integral. It is suitable only if the involved criteria are independent and hence their weighted effects can be added together (Marichal, 2000). There are some interactions between criteria which affect the human expert's inference and cannot be represented by traditional aggregation tools.

There are two nonlinear fuzzy integrals, Choquet and Sugeno integrals, which have been successfully used in literature for aggregation of criteria (Marichal, 2000, and Marichal, 2001).

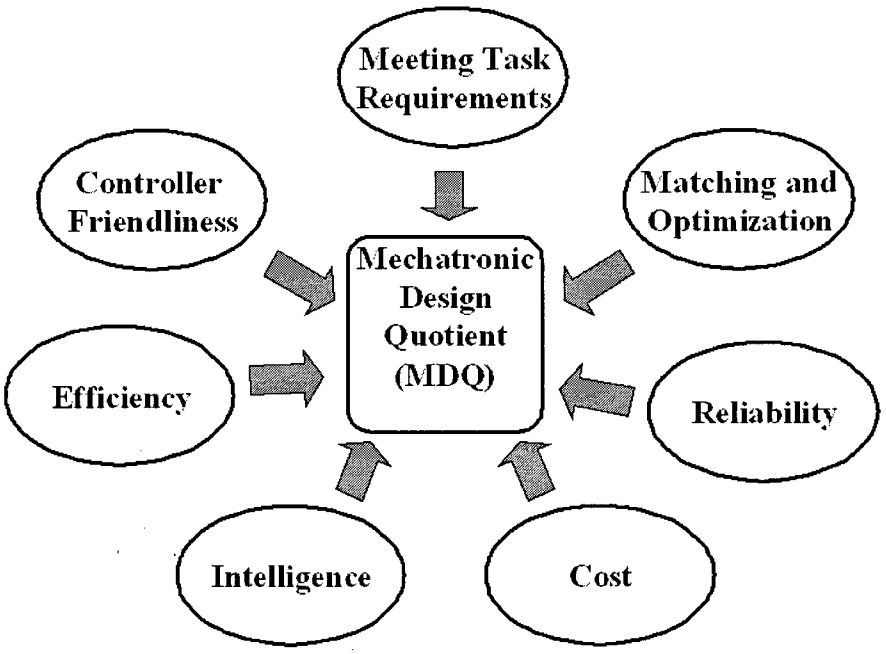


Figure 2.5: Possible attributes of an MDQ.

In particular, Choquet integral fits intuitive requirements for decision making in the case of interacting criteria (Grabisch, 1996; Marichal, 2000; Calvo, *et al.*, 2002).

Considering a finite set of criteria $Y=\{y_1, y_2, \ldots, y_n\}$ in a multi-criteria evaluation problem, the Choquet integral provides a weighting factor not only for each criterion, but also for each subset of criteria. The weighting factor of a subset of criteria is represented by a fuzzy measure in the universe Y satisfying:

- $\mu(\phi) = 0, \mu(Y) = 1$
- $A \subset B \subset Y \Rightarrow \mu(A) \leq \mu(B)$ (2.5)

The useful purpose of defining a weighting factor for each subset of criteria is that interaction between criteria and an expert designer's attitudes can be represented mathematically and can be taken into account in aggregation. As an example, some common semantic interactions and the way they can be represented by fuzzy measures are explained now.

2.3.1 Positive Correlation

Two criteria $y_i, y_j \in Y$ have positive correlation if they have some degree of redundancy; hence, a good score in one of them is usually simultaneous with a good score in the other. For example, for the proposed criteria, reliability and intelligence have positive correlation to some extent. Once a machine is provided with intelligence, for example in the form of self tuning and self diagnosis, these factors result in increased reliability as well. Then these two criteria possess some degree of redundancy. Positive correlation can be modeled by the following inequality (Marichal, 2000):

$$\mu(i,j) < \mu(i) + \mu(j) \qquad y_i, y_j \in Y \tag{2.6}$$

Here $\mu(i,j)$ is the fuzzy measure of the set of criteria which consists of only y_i and y_j.

2.3.2 Negative Correlation

Two criteria $y_i, y_j \in Y$ have negative correlation if it is not common to have high score in both of them simultaneously. In this case, a good score in one of them usually implies a bad score in the other one. For example, cost has a negative correlation with other criteria in the MDQ. High scores in reliability and intelligence for example result in a low score in cost (high cost). Negative correlation can be modeled by (Marichal, 2000):

$$\mu(i,j) > \mu(i) + \mu(j) \qquad i, j \in Y \tag{2.7}$$

2.3.3 Substitution

When two criteria are parallel to each other and are interchangeable (Marichal, 2000), the following substitution property is satisfied:

$$\mu(T) < \begin{Bmatrix} \mu(T \cup i) \\ \mu(T \cup j) \end{Bmatrix} \approx \mu(T \cup i \cup j) \quad T \subseteq Y \setminus y_i, y_j \tag{2.8}$$

2.3.4 Complementarity

This occurs when two criteria are prerequisites (i.e., complementary) of each other to achieve their desired effect (Marichal, 2000). For example 'meeting task requirements' and 'cost' have complementarity to some extent. One has:

$$\mu(T) \approx \begin{Bmatrix} \mu(T \cup i) \\ \mu(T \cup j) \end{Bmatrix} < \mu(T \cup i \cup j) \quad T \subseteq Y \setminus y_i, y_j \tag{2.9}$$

2.3.5 Preferential Dependence

This is the opposite of the preferential independence, which implies the competition between $H(s_i,s_j)$ and $H(s_k,s_j)$ is not affected by the common part y. Here $H(s_i,s_j)$ denotes the aggregated score of partial scores s_i and s_j (Marichal, 2000). Mathematically, one has:

$$H(s_i, s_j) \geq H(s_k, s_j) \Leftrightarrow H(s_i, s_l) \geq H(s_k, s_l) \quad y_i, y_j, y_k, y_l \in Y$$

Here s_i indicates the partial score of a design choice against criterion y_i.

2.3.6 Veto Effect

A criterion y_i has a veto effect if a bad score in this criterion results in a bad global score, regardless of the degree of satisfaction of the other criteria (Grabisch, 1996). Specifically,

$$\mu(T) \approx 0 \quad \text{if} \quad T \subset Y, y_i \notin T \tag{2.10}$$

For example, meeting task requirements has a veto effect on the MDQ.

2.3.7 Pass Effect

A criterion y_i has a pass effect if a good score in this criterion results in a good global score, regardless of the degree of satisfaction of other criteria (Grabisch, 1996). Specifically,

$$\mu(T) \approx 1 \quad \text{if} \quad T \subset Y, y_i \in T \tag{2.11}$$

After specifying the weighting factors for all subsets of criteria, Choquet integral can be used to compute the global score (Calvo, et al., 2002):

$$S = H(S_1, S_2, \ldots, S_n) = \sum_{i=1}^{n} S_{(i)} \cdot \left[\mu(A_{(i)}) - \mu(A_{(i+1)}) \right] \quad (2.12)$$

where (.) indicates the criteria, which should be sorted in ascending order based on their partial scores such that $S_{(1)} \leq S_{(2)} \leq \ldots \leq S_{(n)}$ and $A_{(i)} = \{(i), \ldots (n)\}$ and $A_{(n+1)} = \phi$ (Calvo, et al., 2002).

The main difficulty of the Choquet method lies in the identification of the 2^n coefficients of fuzzy measures. The overall importance of a criterion i is not solely determined by the value of $\mu(i)$. Indeed $\sum_{i=1}^{n} \mu(i)$ is not necessarily equal to one (Calvo, et al., 2002). Intuitive notions expressed in equations 2-6 to 2-11 can serve as a guide here. Another useful concept is the index of overall importance of a criterion, computed by the Shapley value, which is defined as (Calvo, et al., 2002):

$$\Psi(\mu, i) = \sum_{T \subseteq Y \setminus i} \frac{(n-t-1)!t!}{n!} \left[\mu(T \cup i) - \mu(T) \right] \quad (2.13)$$

where $t=|T|$. Then

$$\sum_{i=1}^{n} \Psi(\mu, i) = 1 \quad (2.14)$$

Another useful parameter is the interaction index, which is computed by (Calvo, et al., 2002):

$$\Theta(\mu, ij) = \sum_{T \subseteq Y \setminus i,j} \frac{(n-t-2)!t!}{(n-1)!} \left[\mu(T \cup ij) - \mu(T \cup i) - \mu(T \cup j) + \mu(T) \right] \quad (2.15)$$

A positive correlation leads to a negative interaction index, and vice versa (Grabisch, 1996). The fuzzy measures should be specified in such a way as to satisfy the desired overall importance and the interaction indices.

Difficulties can arise in the specification of fuzzy measures if the number of criteria is high. For example, the seven criteria as mentioned before will form 2^7=128 possible subsets of criteria, which will need the specification of 128 fuzzy measures. Since two of them are predefined (equation 2.5), 126 values should be specified. Specification of these values can be quite challenging because all issues addressed in equations 2.6 to 2.15 should be reflected in these values. To facilitate specification of fuzzy measures, a hierarchical procedure is developed in the present work. It is based on the fact that for three criteria, 6 weighting factors should be specified, and it is not very difficult to come up with a weighting factor for each subset that will reflect the interactions that are present. In the developed method, a hierarchical scheme of criteria is created so that each branch has no more than three sub-branches. The division is such that the items in each category are somewhat similar to each other. By considering this hierarchical pattern, there are only three high level criteria and at most three sub-criteria for

each of them.

The process of specification of fuzzy measures will become simple by this hierarchical procedure. The designer looks at high level criteria and decides about the overall importance of each criteria and the degree of interaction between them. High level fuzzy measures are then specified to meet the viewpoint of the designer. As there are no more then three high level criteria, maximum 6 fuzzy measures have to be specified. For these 6 unknowns, 6 equations can be written to help the calculation of the fuzzy measures so as to satisfy the designer attitudes (3 overall importance and three interaction level). However they can be used only for guidance, because the set of equations is singular and at least one value of overall importance should be established beforehand in order to be able to solve the equations. The same process is repeated for the lower level of criteria. Here the fuzzy measure of each set of criteria expresses the contribution of the set in the high level criterion. If the seven criteria mentioned before are divided into three branches with 3 and 3 and 1 sub-criteria, then, only 18 fuzzy measures are needed to be specified instead of 126, and the specification process becomes more intuitive.

After evaluation of criteria, calculation of the global score also becomes more intuitive and faster. The partial score to each high level criterion is first computed by aggregation of the low level partial scores of its sub-criteria. Then an aggregation between the computed scores of high level criteria gives the global evaluation score of each particular design.

2.4 A New Mechatronic Design Methodology

A systematic design methodology is presented in this section based on the concept of mechatronic design quotient. A product realization can be treated in four stages:

- Conceptual Design
- Detailed Design
- Production
- Final Improvements

The concept of MDQ is used in each of these stages to help the designer in making decisions. Figure 2.6 gives a flowchart of the design approach.

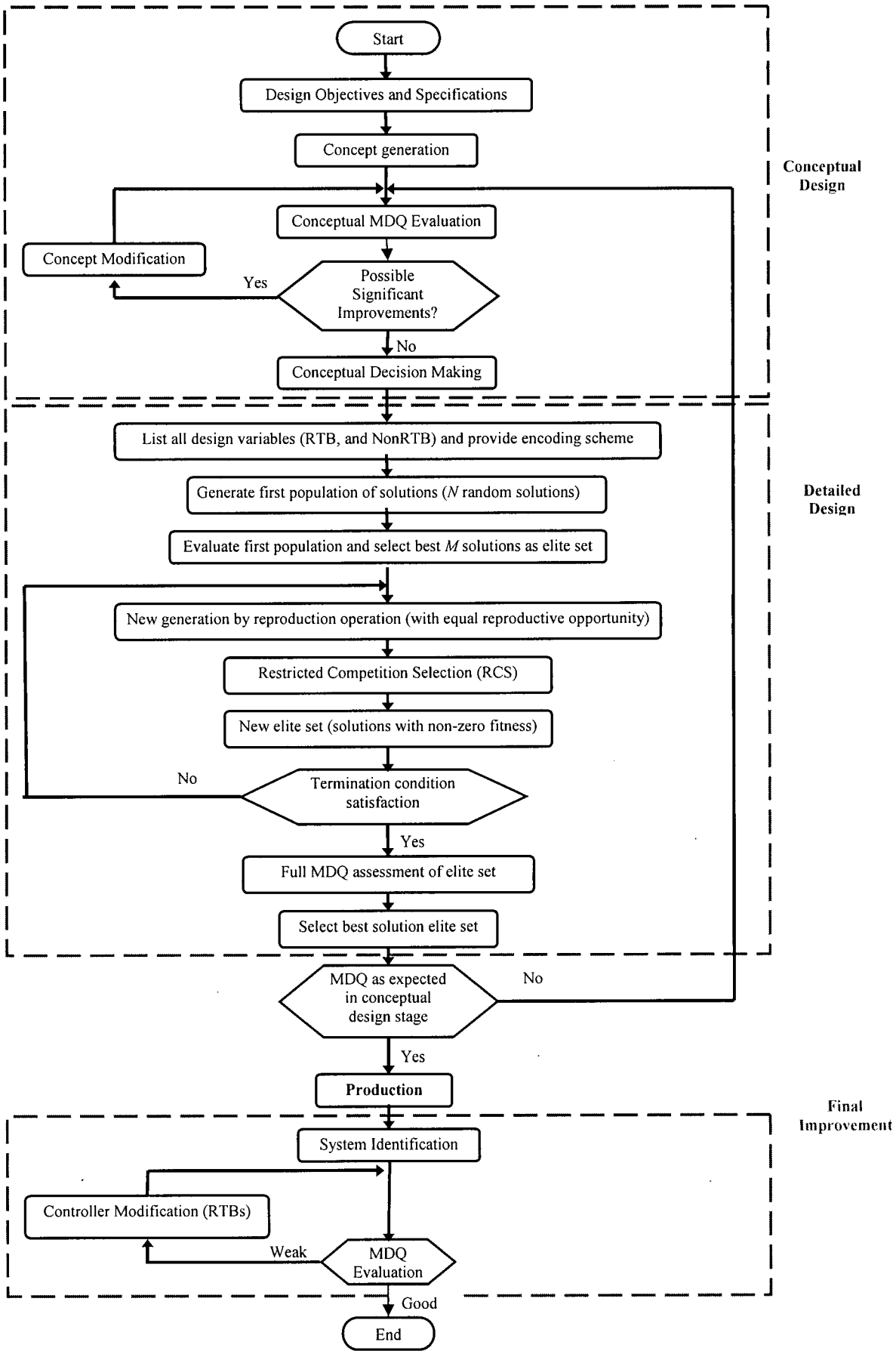


Figure 2.6: Flowchart of the developed mechatronic design methodology.

In the conceptual design stage, MDQ is used as an index to help the designer for proper decision making. The main activities of this stage are conceptual design generation and evaluation and making decision about fundamental structure of the product. Decisions in this stage are basic, but very fundamental. Each decision can have a deep effect on other parts of the system, can significantly change the rest of the design process, and also can change the performance of the product with respect to a wide range of criteria. Because of these considerations, a multi-criteria decision making is desired. The concept of MDQ is used in this stage to guarantee that the overall satisfaction of the product is considered in the decisions. It can also highlight issues that can make significant improvement in the performance of the product from an early stage of the design. Chapter 3 focuses on the conceptual mechatronic design using the concept of MDQ.

In the detailed design stage, the designer lists all the design parameters/variables including both RTP and Non-RTP. Design variables in this stage are either continuous (e.g., controller gains, length of a structural member) or discrete (e.g., DC motor selection, sensor selection). A search universe is then created by listing available options for each discrete variable, and assigning estimated range and possible constraints for continuous variables. An optimization tool is then applied in order to find the solution. As this problem generally does not have a precise solution approach, evolutionary techniques like genetic algorithms are appropriate for the optimization. Since MDQ computation can be complex and time consuming, simple genetic algorithms are not strong enough for this problem. The optimization process is performed in two stages. Niching genetic algorithm is used in the first stage to narrow down the search space to a limited number of elite solutions. A full and comprehensive MDQ competition is performed in the second stage to find the optimum design. Chapter 4 focuses on the detailed design stage.

After the machine is built, the changeability of RTP provides the opportunity for further design improvement. The RTP only affect the RTB of the system, whereas the Non-RTP can affect both RTB and Non-RTB (Li *et al.,* 2001). As the Non-RTP may not be exactly what they should be, the RTB may not be satisfied with the calculated RTP in the detailed design stage. For example, structural properties of a system (e.g., mass, damping, and stiffness matrices) may not be identical to what are calculated and expected in the detailed design stage. As a consequence, desired speed, accuracy and stability (RTB) would not be achieved by the calculated controller gains (RTP). In last design stage, the designer carries out a system identification to practically specify the Non-RTP and then modify the RTP to compensate for any unexpected variations that arise from the detailed design stage. An evolutionary system identification tool is developed in this research by integrating bond graph modeling and genetic

programming, which is explained in Chapter 6.

There also exist techniques to improve the functionality of a machine, such as the consideration of component redundancy, fault detection and diagnosis, and supervisory control. In the final modification stage, the designer considers all controller parameters and possible improvement techniques. MDQ is used to find the optimum controller parameters and to trade off for possible modifications, and decide upon them.

2.5 Case Study: Industrial Fish Cutting Machine

This section describes the industrial machine, on which the developed design and evaluation methodologies are applied. It is a fish processing machine (Tafazoli *et al.,* 1998; De Silva, 2005), known as the "Intelligent Iron Butcher," developed in the Industrial Automation Laboratory of the University of British Columbia. This machine falls under the category of mixed electro-mechanical systems. The developed design methodology will be applied to redesign this machine with the aim to optimize it.

Iron Butcher is used in the fish processing industry to automatically cut the head of fish, which is the first primary stage in the canning process. The goal is to achieve fast and accurate cutting so as to minimize the wastage of useful meat.

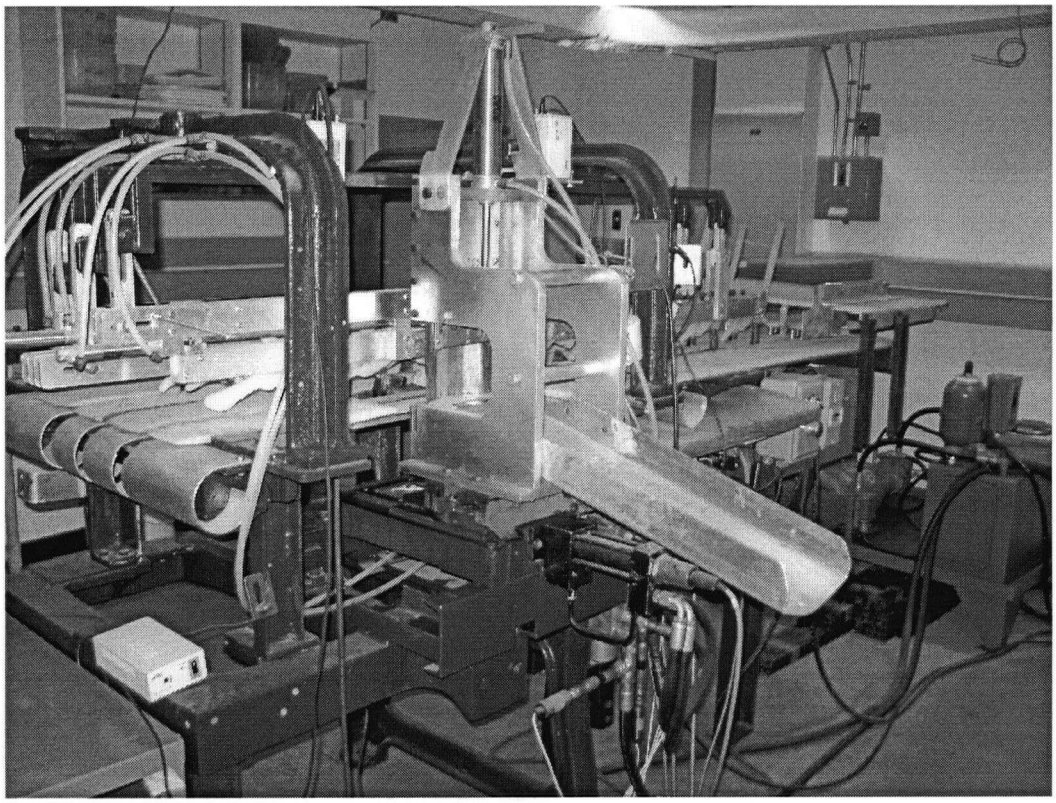
Figure 2.7 shows two pictures of the prototype of this machine. There are four control systems running in parallel, to control the process. They are:

A. Fish motion system

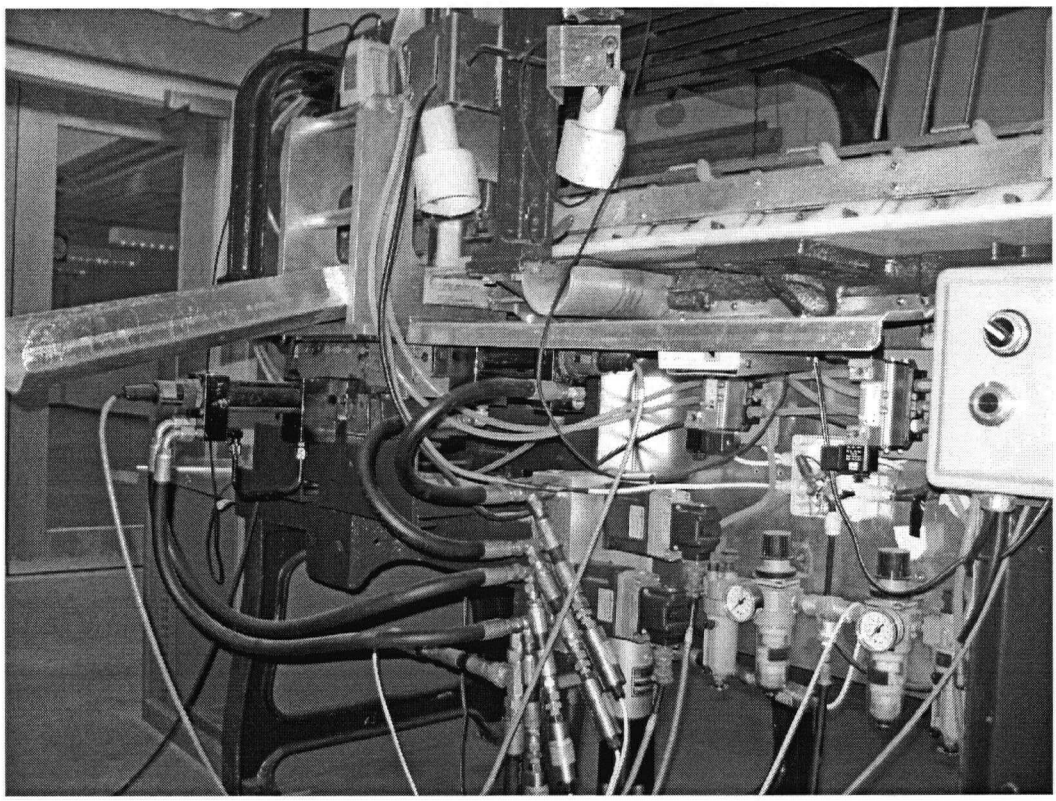
B. Fish stabilization system

C. Cutter vertical motion

D. Cutter horizontal motion (Electro-hydraulic Manipulator)



(a)



(b)

Figure 2.7: The Iron Butcher. (a) A view of the entire machine; (b) Close-up view of the electro-hydraulic manipulator

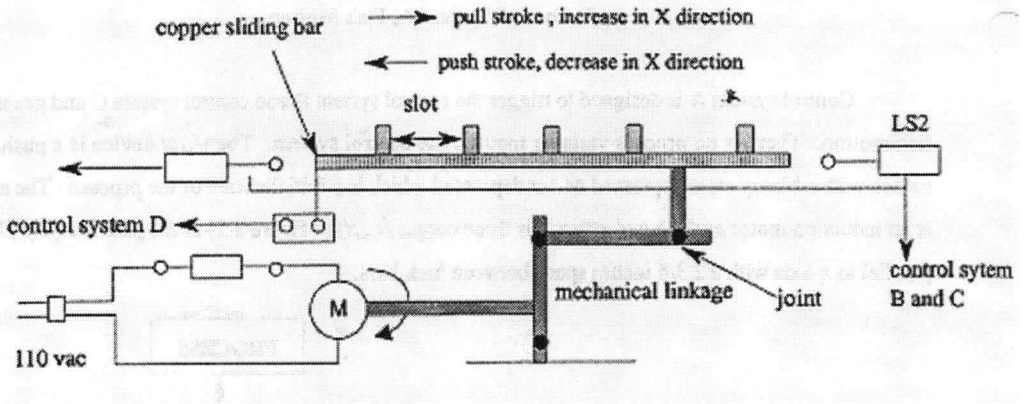


Figure 2.8: Fish motion system (control system A).

2.5.1 Fish Motion System

Control system A is a conveyor system, which accurately moves the fish from the feeder end to the cutting zone. It is a reciprocating, logic-driven system actuated by an AC induction motor. The input device in this system is the push button of the machine, the actuator is an induction motor, and the end effector consists of three copper bars (Figure 2.8). This system also triggers the limit switches to operate systems B and C.

2.5.2 Fish Stabilization System

Control systems B is designed to stabilize the fish while capturing images and carrying out cutting. The input device is a binary limit switch which is triggered by control system A. The end effector is a stabilizing mechanism which applies a force on the fish (Figure 2.9).

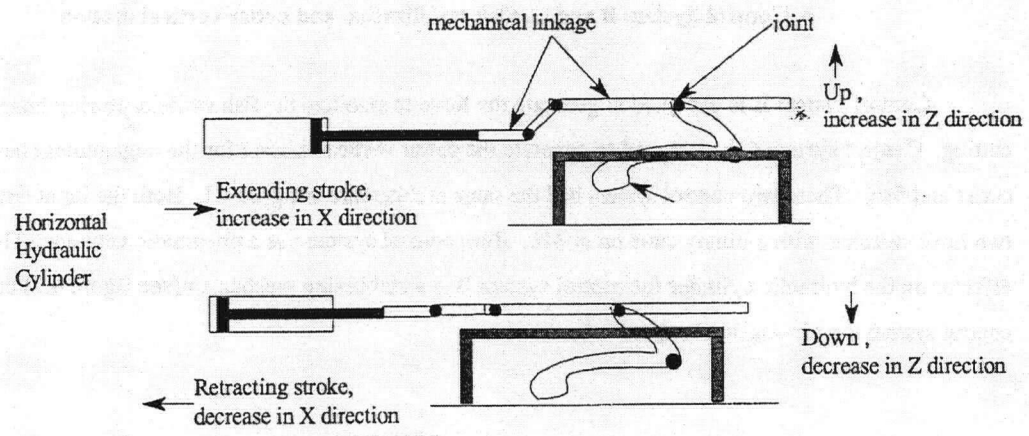


Figure 2.9: Fish stabilizer system (control system B).

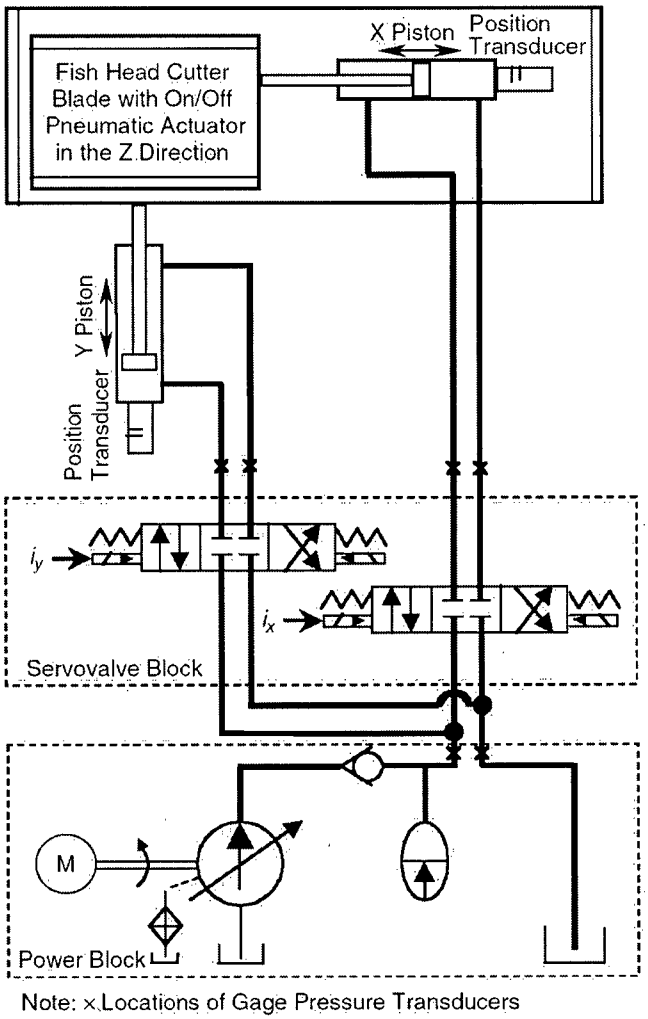


Figure 2.10: Control system of the electro-hydraulic manipulator.

2.5.3 Cutter Vertical Motion

Control system C generates the motion of the cutter to cut the head of the fish. The input device is a binary limit switch triggered by control system A. The end effector is the zig-zag shaped steel cutter which is operated by a pneumatic actuator.

2.5.4 Cutter Horizontal Motion (Electro-hydraulic Manipulator)

Control system D (Figure 2.10) is the most important and complicated part of the system and it is the object of the redesign process here. It is designed to engage the cutter with the desired x-y location (optimum location for cutting) which is determined by a computer vision system using

the characteristic of the processed fish. The on-off signal from the proximity sensor is transmitted to the computer vision system to trigger a CCD camera to capture a 2-D image of the head region of a fish and send the image data to the host computer for processing. A computer vision algorithm processes the image data, obtains the optimum location for cutting, and sends the x-y coordinates as a set point to the control system D. Then the task of system D is initiated, which is to engage the cutter at the set point. The actuator is a hydraulic cylinder which applies a force to the manipulator. There is also a control actuator--a servo valve--which can change the pressure and the flow rate to the hydraulic cylinder. The input signal to the servo valve is the current which determines the applied force to the system by changing the pressure and the flow rate of the hydraulic cylinder. The end effector is the manipulator which is attached to the piston of the hydraulic cylinder.

2.6 Summary

A new mechatronic design methodology was presented based on a multi-criteria design evaluation index called the mechatronic design quotient (MDQ). This design methodology deviates from the traditional sequential design, resulting in improved design performance. The method is not sequential because controller design issues and parameters are treated simultaneously with other issues and parameters. Controllability of the real-time parameters provides an opportunity for further improvement in the system performance after the machine is designed and built, whereas in traditional sequential design approach it is considered an excuse to postpone designing them to a later stage (say, after the machine is built). After presenting a suitable analytical framework for the design methodology, the concept of MDQ was revisited. It is calculated by the aggregation of different design criteria using Choquet fuzzy integral. The developed mechatronic design approach was treated as a multi stage procedure. MDQ was shown to help the designer in all design stages.

Chapter 3

Conceptual Mechatronic Design

The main activities in the conceptual stage of a system design are generation and evaluation of a conceptual design (Moulianitis *et al.,* 2004; Behbahani and de Silva, 2005a; Behbahani and de Silva, 2005d). The designer searches for feasible subsystem choices and conceptual options in such information sources as the Internet, available data on similar machines, catalogs, handbooks, technical reports, and domain experts, to generate conceptual design alternatives. The intention of the conceptual design stage is not to complete a final design, but rather to identify performance-limiting factors of the design proposals in an early design stage with the aim to choose satisfactory specifications for these factors (Coelingh *et al.,* 2002). For a mechatronic system, the conceptual design stage should employ an integrated, concurrent, and system based approach. This stage has several important features, specifically:

- Design decisions are limited mostly to a discrete space, with several limited options.
- Usually there is no explicit analytical relationship or rule to express the preference of a design option over another option, in the decision making.
- Decisions in the conceptual stage are related to the fundamental structure of the final product. Their consequences should be reflected in all the lower level decisions. In practice, if one design parameter is changed from one conceptual option to another, the entire set of design parameters and specifications may change as a result.
- Decisions in this stage affect a wide range of features and specifications. Therefore, a multi-criteria decision making is necessary to select the best option.

In the conceptual stage, the designer should try to generate whatever feasible design choices that may exist for the particular design problem. The generated choices should then be evaluated and compared against the required specifications and desired criteria, with the aim to select the best design. In a mechatronic design, the evaluation of generated concepts should cover all the subsystems in an integrated manner, considering both real-time and non-real-time issues concurrently, and evaluating the overall sense of satisfaction with regard to a set of multiple criteria (i.e., system-based). Because of these features, the designer can use MDQ to facilitate multi criteria decision making in this stage. In addition to its use in evaluation and decision making, MDQ also has the ability to indicate weaknesses of design alternatives which may be improved by some conceptual modifications. The generation, modification and evaluation of a

conceptual design are repeated until the designer is able to justify that the final decision is feasible and the best available one.

3.1 Procedure of Conceptual Design

The main activities in the conceptual design phase are concept generation and evaluation. Practically, the following steps are taken in this phase (Figure 3.1):

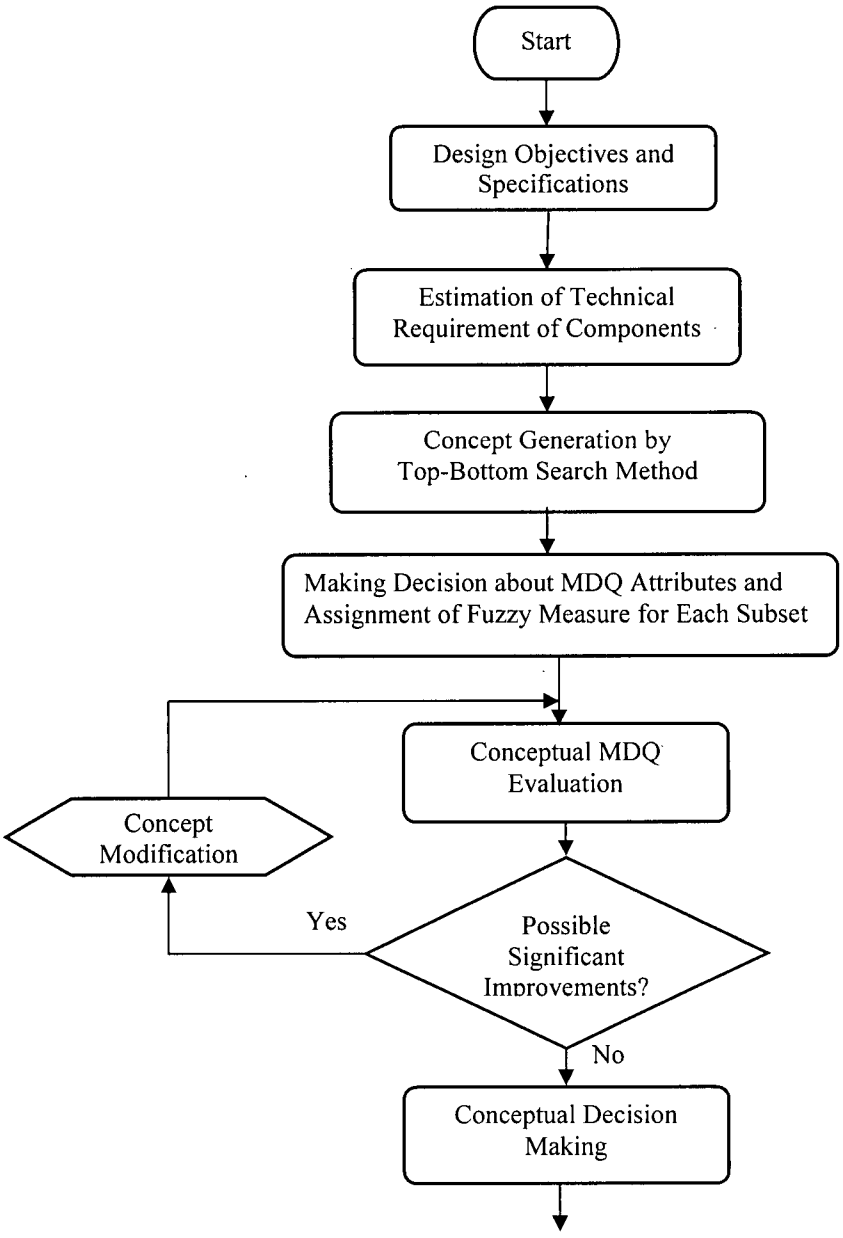


Figure 3.1: Flowchart of conceptual mechatronic design.

1. Review the design objectives, basic requirements, possible constraints, axiomatic design specifications, and the general plan of the system.

2. List the conceptual subsystems or components of the system to be designed using multi-criteria decision making. Many types of subsystems are involved in the design of a mechatronic system, but not all of them will require multi-criteria decision making. Some components can be decided easily with axiomatic requirements. For example, from a practical viewpoint mechatronics is directly applicable to a motion control system. In a typical motion control system, the basic subsystems (components) are the actuator parts, sensor parts, mechanical structure, and the control techniques, expressed as:

$$X = \{x_1, x_2, \ldots, x_n\} \tag{3.1}$$

3. Estimate the technical requirements for each conceptual subsystem, such as capacity, power, bandwidth, accuracy, weight, size, and cost.

4. In this step a top-bottom approach is carried out to generate and appropriately present the conceptual choices. For each conceptual subsystem, list the available and feasible design alternatives which can be called subsystem choices. For example, electrical and hydraulic actuators can represent two choices for the actuator part. Different control techniques can represent the choices for the controller part. Each of these choices may have corresponding sub-choices. All these choices can be presented in a tree-like structure. This step basically includes the concept generation by the help of such means as looking at similar machines available in industry, searching through the Internet and catalogs, consulting with experts, reviewing handbooks, and other sources of information.

5. List all possible and feasible combinations that lead to a design alternative. Not all possible combinations of component choices will require multi-criteria decision making. Some combinations may be clearly weaker than others and can be deleted without evaluation. Retain only those design alternatives that justifiably need a multi-criteria decision making.

6. Decide on the MDQ attributes which are considered important for the particular design. Generally, the seven criteria indicated in figure 2.5 are important in a mechatronic system. Assign a fuzzy measure for each subset of criteria indicating the degree of importance of that subset. Interactions between criteria should be taken into account and can be used as a guide for choosing these fuzzy measures.

7. Evaluate each design alternative, according to all MDQ attributes and assign a partial score for each attribute. However, accurate assessment may not be possible in this stage due to lack of information. For the offered criteria, the following guidelines can be used for design evaluation in the conceptual phase:
   - ***Meeting the task requirements:*** Required bandwidth, estimated required force, available space, and required accuracy are basic attributes that may be used for evaluating this criterion.

   - ***Reliability:*** Basically the number of components in a system has an inverse effect on the system reliability. As a simple estimate, the failure possibility can be considered as the sum of the failure rates of the components in the system. Reliability is then equal to one minus the failure probability. Note that dynamic effects are not taken into account in this simple assessment.

   - ***Intelligence:*** Intelligent features of a machine are incorporated in the subsequent design stages; for example, in the detailed design stage or through programmability of the machine after the machine is built. In the conceptual stage, features such as self calibration, self tuning, self diagnosis, fault tolerance, biologically inspired behavior, human-machine interaction, and having non-model-based conventional control techniques can provide an assessment of the level of intelligence of a machine (Bien *et al.,* 2002).

   - ***Matching:*** Dynamic interactions between components, bandwidth issues, capacity of components and environmental issues can lead to a conceptual assessment of the criterion of matching. In a mechatronic design, a bandwidth matching is required between the frequency content of the desired motion, digital controller frequency (both hardware and software), sampling period, and

bandwidth of sensors and actuators (De Silva, 1989; De Silva, 2005). On the other hand, it is desired that all components of a mechatronic system operate at their optimal capacities. Over-designed and under-designed components degrade the design quality of the system. Some components may not be suitable for a particular environment; for example, mechanical parts of a machine may create an environment which may not be suitable for the proper function of some electrical elements, although in simulation and analysis they may appear to match for that task.

- ***Control friendliness:*** An important attribute in this context is the system nonlinearity, and it can be assessed by estimating the critical nonlinearities of the system; for example, friction. Possible disturbances and system uncertainties, parameter variation, and estimated order of the closed loop system are other issues that may be important in the assessment of controller friendliness in the conceptual stage.

- ***Efficiency:*** This criterion can be evaluated by estimating the probable energy dissipation or wastage in the system, particularly due to friction.

It should be noted that the assessment in the conceptual phase will be tested again in the subsequent stages. However, if a large gap between the estimated MDQ values is found in different stages, the designer should return and review the previous design stages and design assessments again to make sure that the final design is optimal.

8. Aggregate the partial scores by using the Choquet integral to determine the global score of each design alternative.

9. Keeping in mind the MDQ attributes of each design alternative, investigate whether there is possibility to improve its MDQ by some conceptual modifications. Motivated by evolutionary techniques, crossover between good design solutions is a suitable way to search for better conceptual designs. Some architectural modifications such as component sharing, fault detection and diagnosis, and component redundancy can also lead to better design alternatives. If so, consider modifications and go back to Step 6.

10. Select the best design alternative and proceed to the next design stage, which is the detailed design.

3.2 Case Study

A case study using an industrial fish processing machine (the Iron Butcher) is presented now to illustrate the application of the developed methodology of conceptual design. The main steps of the conceptual design for the Iron Butcher are given below:

1-The objective is to design a 2-D positioning table, which is able to move through a maximum stroke of 50 mm in less than 0.4 seconds with an error not exceeding 3 mm. The basic plan is to design a Cartesian table (as for a milling machine) with two motion sensors and two actuators for the two directions (Figure 3.2).

2-The following conceptual components should be designed, or selection decisions have to be made about them:

$X=\{Mechanical\ structure,\ actuator,\ sensor,\ controller\}$

3-The required power, force, and bandwidth can be estimated by estimating the mass, motion trajectory and friction (De Silva, 1989; De Silva, 2005). Table 3.1 gives the estimated values for different conditions.

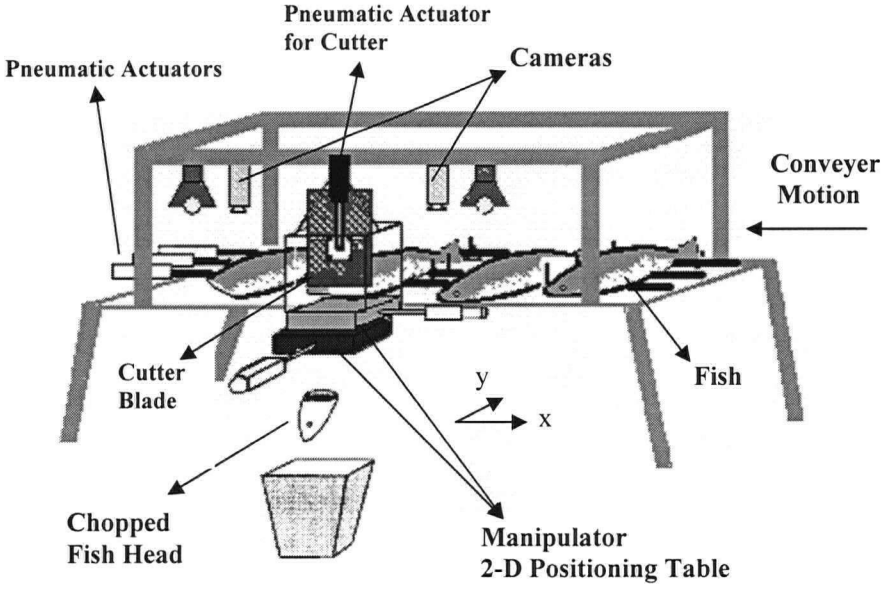


Figure 3.2: Schematic diagram of the Iron Butcher.

Table 3.1: Estimated technical specifications of the machine for different conditions.

Material	Contact	Power (W)	Force (N)	Bandwidth (Hz)
Steel	Direct	1000	1100	20
Steel	Roller	250	320	20
Aluminum	Direct	800	900	20
Aluminum	Roller	200	280	20

4-For presenting the feasible design options for each conceptual subsystem and subsystem choices, a tree-like structure (Figure 3.3) is formed.

5-The main objective of this conceptual decision making process is to decide on the actuator type (electrical or hydraulic), surface contact condition (lubricated, on bearings, or direct contact), and material (steel or aluminum). Other issues do not need multi-criteria decision making and can simply be selected after the above issues are decided upon. For example there are usually embedded position sensors in modern motors and hydraulic cylinders (De Silva, 2005). As the present machine does not have any severe restriction on sensor selection (nearly all sensors meet the required accuracy), the sensor is automatically selected when the actuator is selected.

6-Meeting task requirements, component matching, reliability, controller friendliness, efficiency and cost are important MDQ attributes for this decision making process. These six criteria form $2^6 = 64$ possible subsets of criteria, which need the specification of 64 fuzzy measures. Two of them are predefined (see equation 2-5). Hence 62 values should be specified. It is a rather difficult task to specify these values because all issues addressed in equations 2-6 through 2-11 should be reflected in these values.

For three criteria, 6 weighting factors should be specified, and it is not very difficult to come up with a weighting factor for each subset that will reflect the interactions that are present. In addition, there are 6 unknown values which can be computed by 6 desired conditions (3 overall importance and three interaction level). However it can be used only for guidance, because the set of equations is singular and at least one value of overall importance should be determined in order to be able to solve it. When there are more than

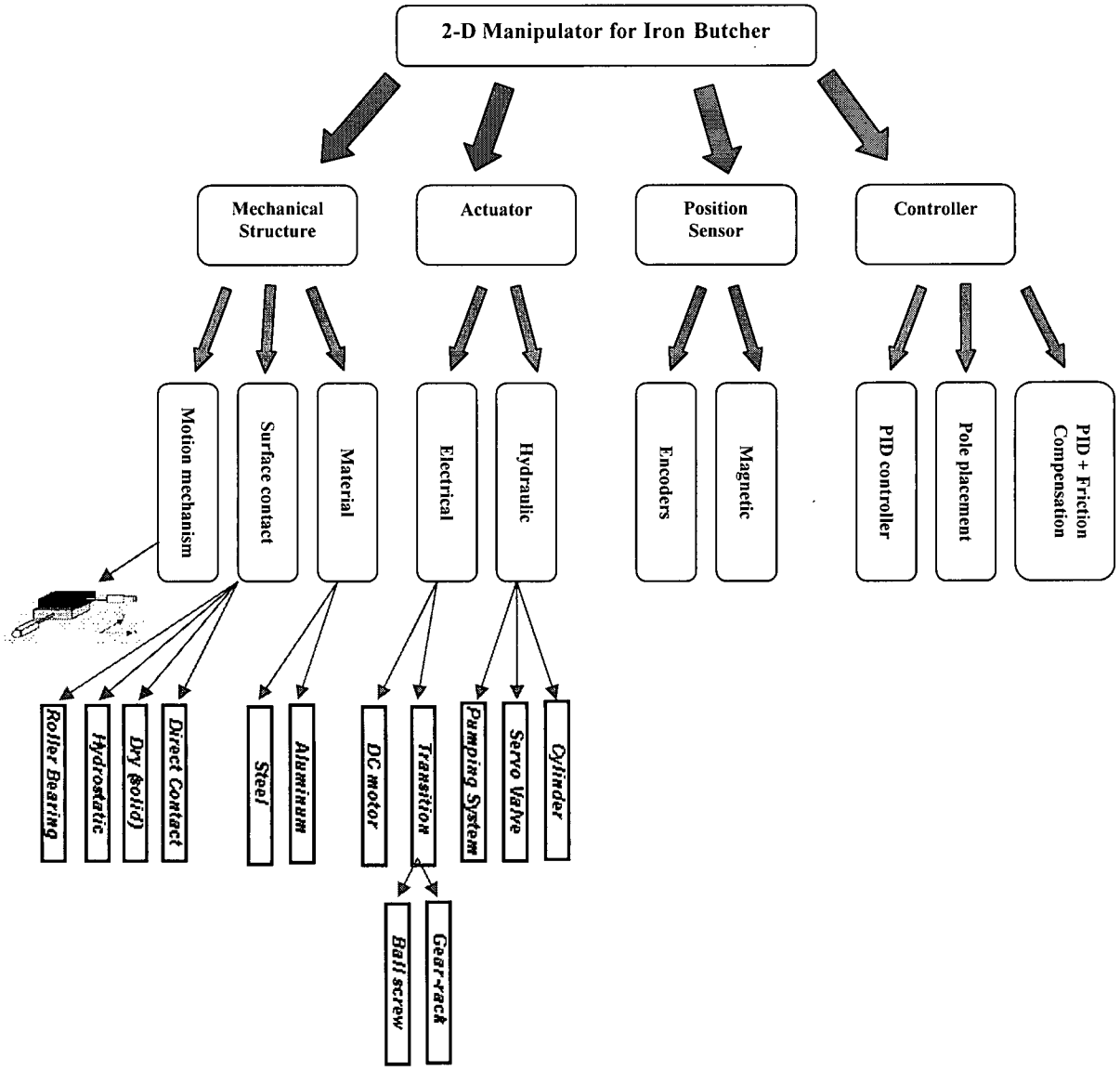


Figure 3.3: Generation of conceptual choices by a top-bottom approach represented in a tree-like structure.

three items, it is rather difficult to come up with a weighting factor of each subset. To overcome this problem, the six criteria are divided into the following three categories:

1- Meeting task requirements; Matching

2- Reliability; Controller friendliness; Efficiency

3- Cost

The division is such that the items in each category are somewhat similar to each other. For example, 'Meeting task requirements' and 'Matching' both have some form of veto effect. 'Cost' has a negative correlation with the other items, and consequently it is isolated. 'Reliability', 'controller friendliness' and 'efficiency' show the quality of the

design. It is good to satisfy them, but a system can be acceptable even if it does not strictly meet these criteria. Now there are only three high level criteria which can be termed 'basic requirements,' 'technical issues,' and 'cost,' respectively. Now it is somewhat intuitive to come up with the weighting effect of each subset of these items. The fuzzy measures given in Table 3.2 are used in the present case study. Using these values, the overall importance of each of the criteria and interaction between them are computed using equations 2-13 and 2-15, as given in Table 3.3. It shows a negative correlation between cost and two other criteria, and also shows a small positive correlation between the categories 'basic requirements' and 'technical issues.'

Table 3.2: High level fuzzy measures used in the Choquet aggregation method. Associated criteria are: 1.basic requirements, 2.technical issues, and 3. cost.

$\mu_1 = 0.52$	$\mu_2 = 0.15$	$\mu_3 = 0.15$
$\mu_{12} = 0.55$	$\mu_{13} = 0.7$	$\mu_{23} = 0.4$

Table 3.3: Overall importance and interaction level for high level criteria including basic requirements, technical issues, and cost.

$\Psi_1 = 0.5317$	$\Psi_2 = 0.1967$	$\Psi_3 = 0.2717$
$\Theta_{12} = -0.035$	$\Theta_{13} = 0.1150$	$\Theta_{23} = 0.1850$

The next step is to repeat the same procedure for the lower level of sub-criteria in each category. First category includes two sub-criteria. It can be said that they have equal importance in satisfying the basic requirements. Therefore, a weight of 0.5 is assigned for each of them for low level aggregation. For the second category, the values given in Table 3.4 are assigned. This provides the results given in Table 3.5. It shows that the criteria have positive correlation with each other.

Table 3.4: Fuzzy measures used for the Choquet aggregation of sub-criteria of "technical issues", including: 1.reliability, 2. controller friendliness, and 3. efficiency

$\mu'_1 = 0.45$	$\mu'_2 = 0.42$	$\mu'_3 = 0.35$
$\mu'_{12} = 0.8$	$\mu_{13} = 0.775$	$\mu_{23} = 0.7$

Table 3.5: Overall importance and interaction level for sub-criteria of "technical issues", including: 1.reliability, 2. controller friendliness, and 3. efficiency

$\Psi'_1 = 0.3842$	$\Psi'_2 = 0.3317$	$\Psi'_3 = 0.2842$
$\Theta'_{12} = -0.0975$	$\Theta'_{13} = -0.0525$	$\Theta'_{23} = -0.0975$

Once the partial scores of a design alternative are found, a low level aggregation should be performed first. The score of 'technical issues' is an aggregation of reliability, controller friendliness and efficiency, based on the fuzzy measures of Table 3.4. The score of 'basic requirement' is the average of 'meeting task requirements' and 'matching.' Then a high level aggregation is performed to determine the global score of a design alternative based on the fuzzy measures in Table 3.2. An obvious advantage of this approach is that it allows to conveniently incorporate the viewpoints of the analyzer by means of weighting factors. In addition, only 14 values have to be specified instead of 62 values.

7, 8- The results of MDQ assessment for some of the design alternatives are shown in table 3.6. The best MDQ corresponds to the choice of a roller bearing, a DC motor with encoder as the position sensor, and a simple PID controller.

9-It was found that some parts of the machine such as camera and all the parts involved in the positioning table are not utilized in half of each cycle, when a fish is pushed into the cutting zone. A new machine can be designed with two fish conveyor systems and with 180 degree phase difference. Then these parts can be shared between them and a design with lower cost (per fish) can be achieved. However, reliability will be less in that case because the number of the components increases, while the overall MDQ of the machine will increase as well.

10- Final decision on the conceptual design stage is to use electrical actuation and incorporate roller bearings for the contact surfaces.

Table 3.6: Conceptual evaluation of MDQ attributes of the conceptual design choices for the Iron Butcher.

	1	2	3	4
Actuator	Electrical	Electrical	Hydraulic	Hydraulic
Contact	Bearing	Hydrostatic	Direct	Bearing
Controller	PID	PID	PID + Friction Compensation	PID
Material	Al	Al	Steel	Al
Meeting Task Requirements	1.00	1.00	1.00	1.00
Matching	0.8	0.6	0.9	0.9
Reliability	0.6	0.5	0.4	0.5
Controller Friendliness	0.9	0.8	0.4	0.8
Efficiency	0.9	0.8	0.4	0.8
Cost	0.9	0.5	0.5	0.7
Overall Score	0.873	0.662	0.704	0.83

3.3 Summary

A practical methodology for conceptual mechatronic design was presented in this chapter. The procedure included a top-bottom approach for the generation of conceptual choices by searching available information sources. This process was presented in a tree-like structure. The generated conceptual choices were then evaluated for comparison against design criteria and for score generation.

The evaluation was based on a multi-criteria design evaluation index called the mechatronic design quotient (MDQ). MDQ contributed to the conceptual mechatronic design by providing a multi-criteria index, with tradeoffs on a variety of criteria involved in mechatronic design, thereby facilitating feasible and optimum decision making. Simplified and practical approaches for preliminary assessment of several important criteria in mechatronic design were presented. With regard to optimal mechatronic design philosophies, the presented procedure was integral, concurrent, and system-based. A practical implementation of the developed procedure was illustrated by presenting the conceptual design of an industrial fish processing machine, which falls into the category of mechatronic systems.

Chapter 4

Detailed Mechatronic Design

Preliminary yet fundamental decisions about the structure and the architecture of a product are made in the conceptual design stage. In the conceptual design, the components and subsystems of the product are specified. For example, it is decided whether the actuator part is electrical or hydraulic. The control technique is also selected and only its parameters remain to be calculated. Once the overall structure of the product is designed, the designer should calculate the corresponding design parameters, or choose the corresponding components from available options in the market. This stage of design involves computation and specification of design parameters; hence termed detailed design.

Some of the design parameters are changeable and controllable even after the machine is built and some others are not. Some parameters are continuous and should be computed and others are discrete and are limited to a finite number of available options. Regardless of these classifications, all design variables should be computed and optimized in a concurrent and integrated manner with respect to multiple criteria which are important in the performance of the product. In other words, a mixed mechatronic system ideally needs an integrated, concurrent, and system-based design approach.

Complicated design of a mixed system can be treated as an optimization problem by using a proper design evaluation index. In the design approach developed in this thesis, a mechatronic design quotient or MDQ (De Silva, 2003) serves to evaluate the fitness of design trials in an optimization process. The optimization process is performed in two stages because of the complexity of the problem. In the first stage, niching genetic algorithm is used to find local and global optimal design alternatives with respect to some essential MDQ attributes. In the second stage, these local optima will compete with each other with respect to all criteria involved in MDQ. This design methodology offers a concurrent, integrated, and multi-criteria approach, which will provide a mechatronic design that is optimal with respect to the design criteria included in the MDQ (Behbahani and de Silva, 2006a, b).

4.1 Niching Genetic Algorithm

Genetic algorithms represent an evolutionary optimization method which mimics the evolution of biological species in nature. It is a stochastic search method based on Darwin's principle of

survival of the fittest. In this method, design variables of a potential solution are encoded as a simple chromosome-like data structure. The encoding is usually performed by representing each design parameter in the binary form, which results in a string of 0s and 1s. This representation of each trial solution is considered a chromosome and each binary value is considered a gene. First generation of trial solutions is created randomly; hence, there is usually no acceptable solution in it. Next generation is created by applying the following reproduction operations on the current generation:

1- *Crossover:* two parent chromosomes are selected and a gene branch of one is exchanged with the corresponding gene branch of the other; hence, two new children are created.
2- *Mutation:* one chromosome is selected and a gene branch is replaced by a randomly created gene branch.
3- *Survival (selection):* the best solutions are copied into the next generation and the rest are discarded.

There are many strategies for applying a genetic algorithm, but all of them have similar features. In order to mimic the natural evolution of biological species, selection of the parents for reproduction operations and the place in the chromosome for applying the particular operation are all performed somewhat randomly, but in such a way that a trial solution with a higher fitness has a higher reproductive opportunity. Due to this strategy, the next generation would be probabilistically better than the current generation. By continuing this process, the fittest solution will naturally converge to the optimal solution.

There is a problem associated with implementation of genetic algorithms for optimization of MDQ. Fitness evaluation of trial solutions is the key procedure in the genetic algorithms, which is repeated many times. It is very important that this fitness evaluation be fast and does not require complicated analysis. A general and appropriate representation of MDQ should include a wide range of criteria. The accurate assessment of these criteria usually needs rather complicated analysis, which can be time consuming and costly. In particular, the main objective of the developed design approach is to provide a comprehensive and multi-criteria view to the evaluation of design trials in the design process. This, however, conflicts with the efficient utilization of optimization approaches. A comprehensive evaluation would be too time consuming and complicated to be performed in the course of a reasonable optimization process.

To counter this problem, it is proposed to perform the optimization process in two stages. In the first stage, only a combination of essential and more important criteria – those which have a veto effect on the MDQ assessment – is applied for fitness evaluation. In addition, a simplified

and approximate evaluation is performed in this stage in order to save time and cost of computation. In contrast, the optimization process is implemented in such a way to retain not only the global optimum, but also local optima which could potentially represent better designs if a full and accurate assessment were applied. These optimal design trials can be considered as candidate solutions to be optimum. The variations of evaluation indices due to the change of design parameters are expected to be rather smooth in a real physical system. Hence different optima are expected to have significant differences from each other, representing different possible configurations for the system. By retaining all optimal solutions, each possible configuration is allowed to represent its elite solution for a final competition, even if it has rather low fitness in comparison to other configurations. In the second optimization phase, a full and accurate MDQ assessment is performed as a competition between candidates to find the global optimum design.

One of the desirable features of a simple genetic algorithm is its ability to escape local peaks due to its random variations. It means that even a solution corresponding to a local maximum will not likely survive if its fitness is significantly less than the solutions near the global peak. Even in the case of approximately equal peaks, a simple genetic algorithm will randomly converge to one of the peaks, because it does not have any control on the competition between different peaks. This desirable feature of simple genetic algorithms makes them unsuitable for the particular problem considered here, where the designer is interested to find local peaks as well as the global peak. In other words, simple genetic algorithms are not suitable for optimization of a multi-peak function when the designer is interested to know all the peaks.

To address this problem, the present work proposes to use niching genetic algorithms. The niching genetic algorithms represent a branch of evolutionary algorithms dedicated for multi-objective and multi-modal optimization problems. They are used when the function to be optimized has several peaks and the designer is interested in all of them, regardless of whether it is a local peak or the global peak of the function (Cho *et al.,* 2001, Kim *et al.,* 2002, Himeno *et al.,* 2003).

Analogously to other terms derived from genetic science of biological species, in biology niche means a unique ecological role, location or job of an organism in a community for which a species is well suited within its community, including its habitat, what it eats, its activities, and its interaction with other living things. The niche of an organism permits it to survive in its environment. In other words, the particular task, formation and resources of an ecosystem reduce the interspecies competition with other ecosystems which in turn helps a stable survival

of this ecosystem. Different members of an ecosystem, however, undergo competition with each other to survive, resulting in a gradual evolution of the ecosystem.

In a niching genetic algorithm, a collection of solutions with similar configuration is considered as an ecosystem. Several techniques are available to implement a niching genetic algorithm, including sharing, deterministic crowding, and restricted competition selection (RCS) (Himeno *et al.,* 2003). The RCS technique is the best technique for the specific problem of the present work, because other techniques maintain many solutions around each niche. In this particular problem only the best solution is needed per niche, because individuals in a niche have similar characteristics. The RCS approach has also better matching with the mystery of the survival of the ecosystems in nature. In this approach, the particular configuration of a collection of solutions with similar configuration (ecosystem) reduces the chance of competition with the solutions with different configurations. Two solutions will compete with each other only if their difference is less than a threshold, called *niche radius*. By restricting competition among dissimilar individuals and performing competition only between similar individuals, a stable subpopulation will find an opportunity to be formed around a local optimum. Due to competition between similar individuals, each subpopulation evolves and converges to its best. The best solution in a niche represents the elite of that subpopulation. The set of elite solutions is finally created which represents the best solutions with different possible configurations. The corresponding procedure of this technique is as follows (Cho *et al.,* 2001):

Step 1- Generate N random trial solutions and evaluate their fitness. Select M fittest solutions as an elite set.

Step 2- Create N new random solutions by applying reproductive operations on the previously created generation. Note that, unlike in a simple genetic algorithm, in the RCS niching algorithm you do not have to consider any discrimination in reproductive opportunity between individuals.

Step 3- Add the elite set to this population to generate a competing set with $N+M$ solutions.

Step 4- This is the fundamental step of RCS. Confront each trial solution to all other trial solutions. If the difference between them is less than the nominal difference (niche radius), perform a competition between them and set the loser's fitness to zero.

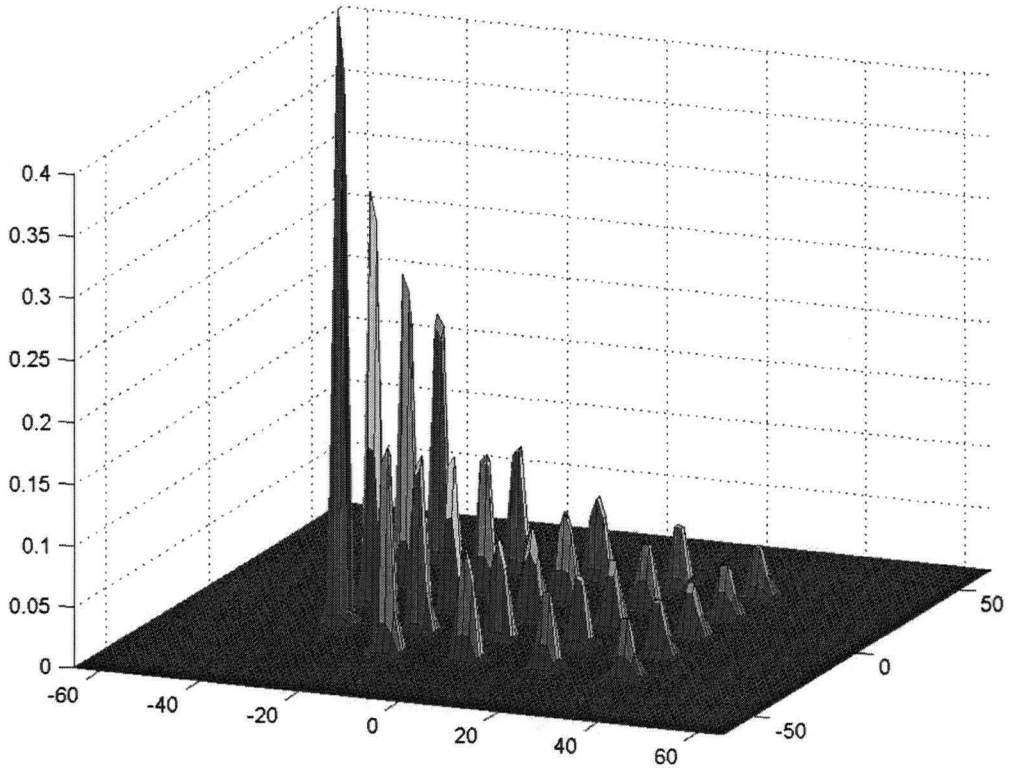
Step 5- Generate the new elite set as the set of solutions with non-zero fitness.

Step 6- Randomly select N solutions from the competing set to be the parents for the next generation. It is desirable to include the elite set in this population.

Step 7- If terminating condition is not satisfied, go to step 2; else, stop the process.

Based on the described RCS procedure, an optimization tool has been developed for multi-peak functions. To verify the developed tool, it has been tested on several multi-peak functions, for example the function shown in Figure 4.1. The obtained results are quite satisfactory with respect to the number of the peaks found by the tool and their accuracy.

In the developed method, a niching genetic algorithm is used in the first optimization stage, not to find the optimal solution but to limit the search space to some optimal candidates. In the



$$F = 500 - \frac{1}{0.002 + \sum_{i=1}^{24} \frac{1}{1+i+[x-a(i)]^6+[y-b(i)]^6}} \qquad a(i) = 16[\mathrm{mod}(i/5) - 2]$$

$$b(i) = 16[fix(i/5) - 2]$$

$$-65 < x, y < 65$$

Figure 4.1: An example of a multi-peak function used to verify the developed optimization program by a niching genetic algorithm (Himeno *et al.*, 2003).

second optimization stage, the designer is able to add other criteria into MDQ assessment and make an accurate analysis to make a practical competition between these optimum candidates and find the real optimum.

4.2 MDQ Optimization

Figure 4.2 shows the flowchart of the design methodology developed in the present research, based on MDQ optimization. The conceptual design stage is presented in Chapter 3. The following steps are taken in the detailed design stage:

1. List all essential subsystems and component which should be designed. Each of these components may have sub-components. If all these sub-components are arranged together, a tree-like structure is formed. For each sub-component or design parameter in the lowest level of this tree, decide whether it needs a concurrent, integrated and multi-criteria design methodology. Not all the components and variables in a mechatronic system need to employ a multi-criteria and integrated design approach. For instance, some components do not impose any limitation on the performance of the entire system or do not have considerable interaction with other components in the system; hence, they can be designed or selected separately. List all the design parameters determination of which justifiably will need a concurrent design approach, due to their interaction with other parts of the system.

2. For each component in the list, decide whether it should be built or selected from available options in the market. If it has to be built, list design variables which specify the structure of that component and provide a rough estimation of the range which is reasonable for each design parameter. If it has to be purchased from available options in the market, provide a rough estimation or a reasonable range of technical requirements for that component, and prepare a list of available options in the market by searching information sources such as Internet, and manufacturer's catalogs.

3. List all design parameters which can describe a trial design solution for the system, and develop an encoding scheme to represent each trial design by a string of binary values. This means that each design parameter is discretized to a finite number of segments, expressed as a power of 2. The discretization process is not particularly a problem for a

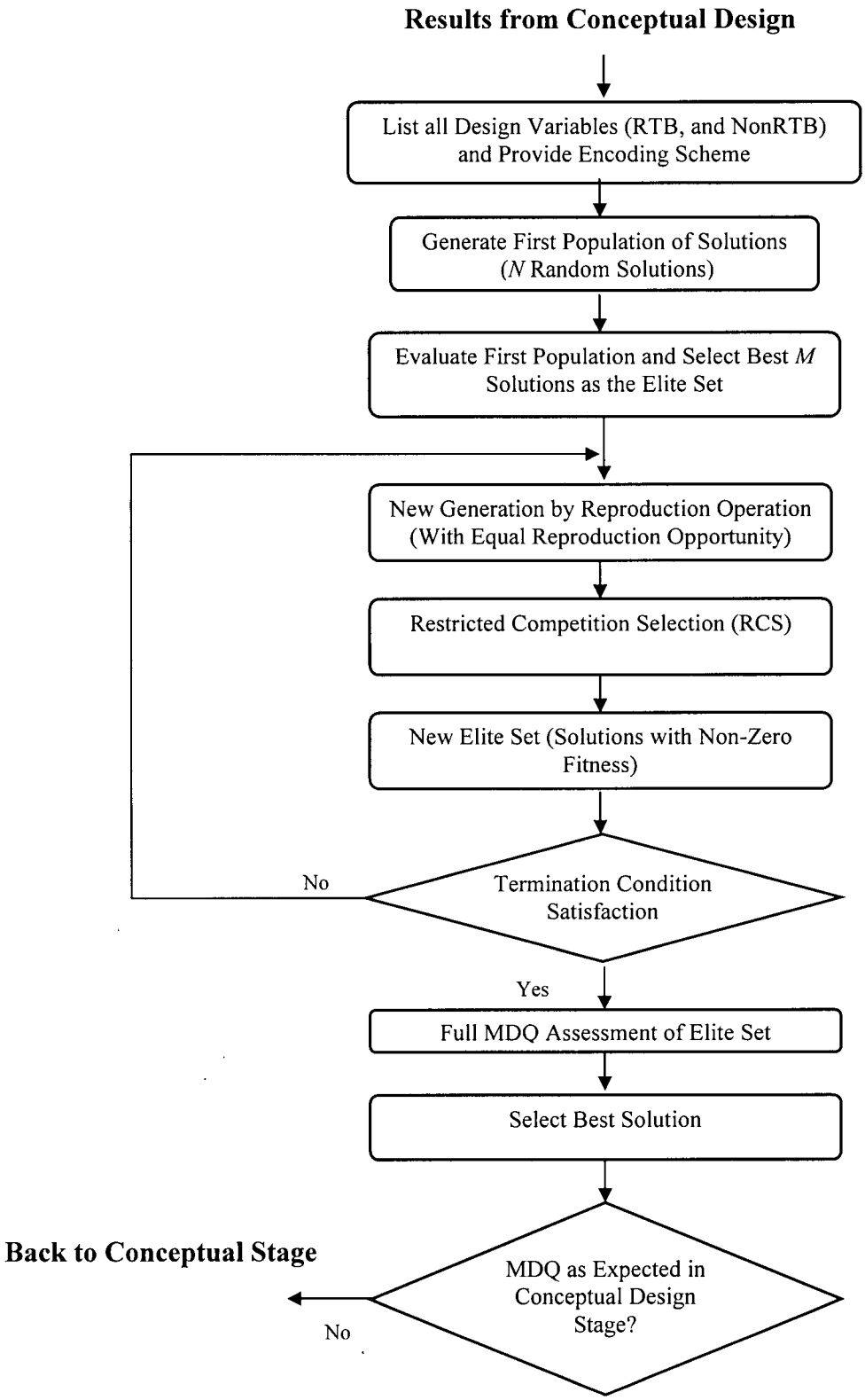


Figure 4.2: Flowchart of detailed mechatronic design.

continuous parameter as far as it provides enough bits to achieve the desired level of precision in the feasible range of that particular parameter. For a discrete parameter, for

example, the case where a part is selected from available options in the market, the discretized value refers to the order of each option in the set of options. If the number of available options is not a power of 2, then some unnecessary bit patterns will exist. In the genetic algorithm process, a trial solution which addresses to one of these extra values can be discarded, or some parameters may be represented twice so that all binary strings result in a legal set of design variables.

4. Decide on the MDQ attributes which are considered important for the particular design. Generally, the seven criteria indicated in Figure 2.5 are important in a mechatronic system. For the first stage of optimization, consider only those criteria that are more important. Specifically the criteria which have a veto effect or a pass effect on the evaluation of the system should be considered for this optimization phase.

5. Provide simple and fast routines to analyze each trial solution with respect to the above criteria. In this stage, the designers may need to use computer software or develop their own programs to model the system.

6. Provide a relationship or develop a small program which can evaluate how two different trial solutions are related, and decide if they are considered as two different configurations or not.

7. The stage is now set to use the niching genetic algorithm to find local optima which are called elite solutions. The flowchart of the niching genetic algorithm with restricted competition selection (RCS) is explained in section 4.1. Use this strategy and determine elite solutions which have fairly high fitness levels. The first stage of optimization is finished here and now the designer should perform a more detailed and comprehensive evaluation between these elite solutions to find the best one, which can be claimed as the optimum design.

8. Consider all the MDQ attributes and assign a fuzzy measure for each subset of criteria indicating the degree of importance of that subset. Interactions between criteria should be taken into account and can be used as a guide for choosing these fuzzy measures.

9. Evaluate each elite design according to all MDQ attributes and assign a partial score for each attribute. In this stage, the designer may need to develop computer programs and use available simulation tools for the design tasks. The analysis in this stage should be as accurate as possible. It is not a critical problem if the run-time of the simulations is high because it is not repeated many times.

10. Aggregate the partial scores by using the Choquet integral to determine the global score of each elite design. The design with the highest global score is considered as the optimum design.

4.3 Case Study

Detailed design of the manipulator of an industrial fish cutting machine (Figure 4.3) is presented here. Conceptual design of this system by using the concept of MDQ has been described in Chapter 3. On that basis, it has been decided to use DC electrical motors for actuation, aluminum as the material of the structure of the carriages, and use PID control for positioning of the cutter module of the machine. This section focuses on the next design stage--the detailed design--and explains the process of application of the developed method on the detailed design of this system.

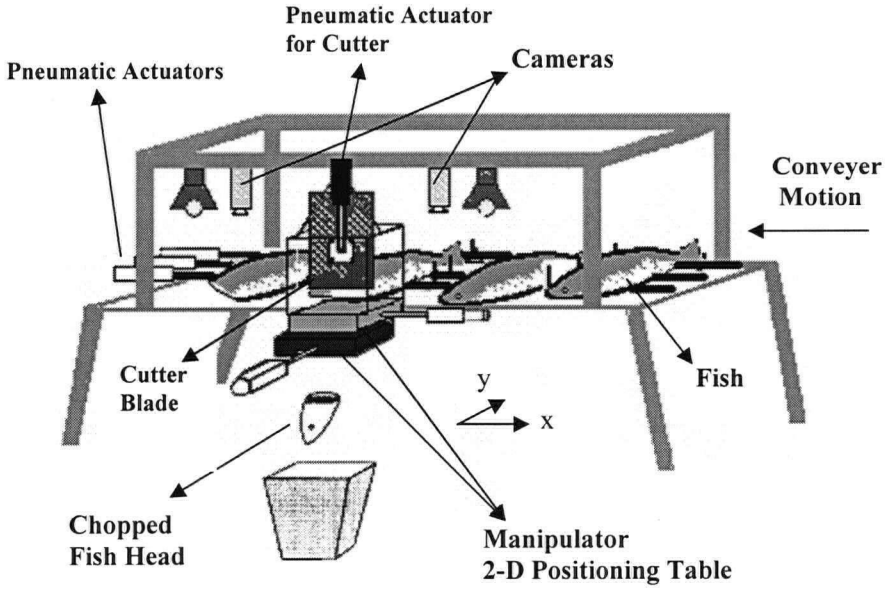


Figure 4.3: Schematic diagram of the Iron Butcher.

The main steps of the detailed design are given below:

1. Not all components and subsystems of the machine need multi-criteria and integrated design. For example, the sensory part does not have considerable interactions with other parts and also does not impose any technical limitation on the performance and the capacity of the system; hence the sensor parts can be selected separately based on some criteria such as price, and matching with the environmental conditions, after other parts are designed. In addition, most new actuators come with an embedded sensor, which facilitates the design and the production of the machine. The main objective of the detailed design stage for this machine is to select a proper motor, transmission system, bearings, and the structure of the carriage, and design an optimal controller for it, all in a concurrent manner. Other parts of the machine do not have much interaction with these essential components and can be selected or designed separately after these essential parts are designed. Therefore, the design parameters are the motor, transmission, bearing, dimensions of the rectangular plate, and controller gains in x and y directions.

2. The motor, transmission system, and bearings are acquired from available options in the market. The carriage has to be designed and fabricated separately. The controller has to be designed and then implemented in a proper digital control platform. Required technical specifications of the motor, transmission system, and bearings are roughly estimated here. In order to find the maximum speed and acceleration, which may be needed for the machine motion, a simple second order system is considered. Settling time and damping ratio are taken as 0.4 seconds and 0.7, respectively. Then, a step input is applied to this system so that the steady state output of the system becomes 5 cm. It has been found that the maximum speed and acceleration are 0.32 $^{m}/_{s}$ and 6.1 $^{m}/_{s^2}$, respectively.

   The weight of the cutter is approximately 10 kg and its intersection size is approximately 10*10 cm. The size of the plate can be exactly equal to this values and its thickness is roughly estimated to be between 2 to 20 mm. Based on this rough estimates and a reasonable safety factor, the required power for the motor is estimated to be between 100 and 200 W.

   Based on the estimated required power and by searching pertinent web sites through the Internet, 16 different DC motors are selected. Some of them are simple DC rotary motors which needed a transmission system (e.g., ball screw unit) to convert rotary motion to

linear motion. Some others are linear actuators which are formed by DC rotary motors with a ball screw or roller screw transmission system embedded in them. There are linear DC motors as well, which directly provide linear motion.

Eight different transmission systems are also selected which are ball screw and roller screw units with different values for lead rate, load capacity, and efficiency.

3. There are seven design variables including the thickness of the plate, controller gains (three gains), motor, transmission, and bearings. They are encoded as a binary string to be optimized by the genetic algorithm.

4. 'Meeting task requirements' and 'matching' are two criteria which have veto effects on the evaluation of a design trial, and are considered as essential criteria for the first stage of the optimization.

5. In assessing the 'meeting task requirements' criterion, a SIMULINK model is developed to analyze the performance of the controller. Rise time, settling time, overshoot, and steady state error of the response of the system are compared with ideal response specifications and a partial score is assigned describing the degree of satisfaction of the 'meeting task requirements.'

   Three different issues are analyzed to assess the 'matching' criterion, including:

   - Component capacities
   - Bandwidth issues
   - Stress and deflection limitations

   First, it is checked if the motor works near its nominal capacity. Basically, this refers to the fact that under-designed and over-designed components are not desirable in a good mechatronic system. For each trial design, the maximum speed and maximum force imposed to the motor are found from the response of the SIMULINK model. A partial score is then assigned to each design trial describing how the capacity of the motor has benefited in the particular design trial.

   The next issue that is analyzed to assess the matching criterion is the bandwidth. Basically, the dominant natural frequency of the system should be several times larger than the frequency content of the control action. For each trial design, the natural

frequency of the rectangular plate is calculated approximately and compared to the bandwidth of the motion, and a partial score is assigned to describe the degree of satisfaction of this aspect.

Stress and deflection limitations are also addressed in the assessment of the 'matching' criterion. The stress in the plate due to its own weight, the weight of the cutter and the vertical cutting force should be several times less than the yield stress of the aluminum. The deflection of the plate under these loads should be smaller than a desired limit which is considered to be 1 mm in the present system. For each trial solution, the maximum stress and maximum displacement in the plate are computed by some simplified elasticity equations and then a partial score is assigned to each trial solution describing the degree of satisfaction of these issues.

The partial score of each trial solution with respect to the 'matching' criteria is considered as a linear aggregation of capacity matching, bandwidth issues, and stress-deflection limitations. The fitness evaluation of each trial solution is then a linear aggregation of the partial scores to the 'meeting task requirement' and 'matching' criteria.

6. The difference between two trial solutions can be calculated either in genotype or phenotype presentation of them. In this particular application, there are some discrete design parameters which refer to the order of a part in a list of available options. The genotype representation of these parameters does not have a physical meaning; hence, genotype comparison is not appropriate. In the design of the machine, the degree of discrepancy between two solutions is evaluated by comparing all the parameters that are affective in the response, even those parameters that do not appear in the genetic algorithm representation of solutions. For example, all electrical and mechanical specifications of motors are considered in this calculation.

7. Niching genetic algorithm with restricted competition selection (RCS) is used to find local optima which are adequately separated from each other to be considered as different configurations. Four elite solutions are found, which correspond to different possible configurations. Table 4.1 presents s a brief description of these four elite solutions.

8. Now that the elite solutions have been found, a full and comprehensive competition should be performed between them to find the best design. Meeting task requirements,

matching, reliability, controller friendliness, efficiency and cost are important MDQ attributes for this decision making process. These six criteria form $2^6 = 64$ possible subsets of criteria, which need the specification of 64 fuzzy measures. A hierarchical approach has been developed to facilitate this task, as described in Chapter 3. The same procedure and fuzzy measures are employed in this stage.

Table 4.1: Elite design trials found by the RCS niching genetic algorithm.

		Elite #1	Elite #2	Elite #3	Elite #4
Motor Specifications	Motor Type	Rotary DC Motor	Linear DC Motor	Rotary DC Motor	Linear Actuator
		AEROTECH BMS-60	AEROTECH BLMUC-111	AEROTECH 1050	
	Maximum Load Capacity	1.68 N.m	209 N	5.22 N.m	173 N
	Maximum Speed	10000 RPM	10 m/s	5000 RPM	10 m/s
Transmission	Type	Roller Screw	N/A	Ball Screw	N/A
	Pitch (mm/rev)	12.7	N/A	2	N/A
	Efficiency	60%	N/A	80%	N/A
	Plate dimensions *(mm)*	14.5	11.4	48.4	14.5
	Integrator Controller gain	167.96	163.49	36.07	162.7451
	Proportional gain	1213.7	1327.5	805.8824	1041.2
	Derivative Controller gain	17.1373	115.8039	27.1373	94.1961
	Settling Time	0.3604	0.3825	0.3457	0.3819
	Overshoot	0	0	0	0
	Steady State Error	.4	0.25	0.11	0.1577
	Maximum Voltage	60.8	66.37	40.3	52.1
	Maximum Load	1.2345	109.31	3.98	84
	Maximum Speed	0.4360	.4590	0.4	0.4439

Table 4.2: MDQ evaluation of elite designs.

	Elite # 1	Elite #2	Elite # 3	Elite #4
Meeting Task Requirements	0.996	0.993	0.988	0.999
Matching	0.7	0.7	0.66	0.7
Reliability	.9	0.8	0.8	0.8
Controller Friendliness	0.96	0.87	0.98	0.84
Efficiency	0.49	0.06	0.49	0.04
Cost	.9	0.5	.7	.8
	0.8544	0.686	0.7679	0.7852

9. Table 4.2 shows the results of evaluation of these 4 solutions with Choquet fuzzy integral.

10. It is clear that first design is the best and can be considered as the optimal design.

4.4 Summary

A new mechatronic design methodology was developed based on the optimization of mechatronic design quotient (MDQ) using a niching genetic algorithm. This design methodology offers an integrated, concurrent, and system-based viewpoint to mechatronic design, which deviates from the traditional sequential design methodologies. The optimization process is performed in two stages. First stage is carried out by using a niching genetic algorithm. It allows different possible configurations to present their elites to the final competition. Restricted competition selection (RCS) strategy allows the creation of a stable population around each niche even if it is quite weaker than the other solutions, and presents its elite into the set of elites. It is well-matched for the particular application discussed in this chapter, because the designer is interested in keeping all possible configurations for the final competition, and on the other hand, he/she is interested in only the best solution in each subpopulation. Other solutions in the sane niche have similar configurations and need not be kept.

The evolutionary optimization technique will probabilistically converge to the solution which has the best MDQ rating. A wide range of design attributes can be suitably integrated into MDQ by using Choquet fuzzy integral to adequately express the designer's attitudes about the overall satisfactory performance of the entire system. Integration of different design attributes, along with the use of an intuitive aggregation technique to express dynamic interaction between criteria, and application of an evolutionary niching genetic algorithm provide a mechatronic design tool which benefits from all opportunities to provide the best design with the highest global sense of satisfaction of the final product.

Chapter 5

Bond Graph Modeling

A mechatronic system is a mixed system, containing different components that belong to different domains such as mechanical, electrical, hydraulic and control. The schematic diagram of a typical mechatronic system is shown in Figure 5.1. Basically there is a dynamic process plant whose dynamic behavior has to be controlled to achieve a desired response. This figure shows the main subsystems/components that are seen in a mechatronic system. They are interconnected through the flow of either information or power. The entire system can be divided into two domains:

- High-power domain (power domain): some components of the system operate under high power. Their connections to other parts are also associated with high power/energy transfer. The variation of the power variables is a very important issue, which must be analyzed in this domain.
- Low-power domain (information domain): components in this domain of the system are connected together through low-power energy transfer links. This low-power energy transfer is needed in the transformation of important information about the system. The low-power energy transfer does not have a significant effect on the overall energy transfer in the system and can be neglected in comparison with the energy transfer in the power domain. However, the information transfer associated with it is the main aspect of this domain, which has to be considered in system analysis.

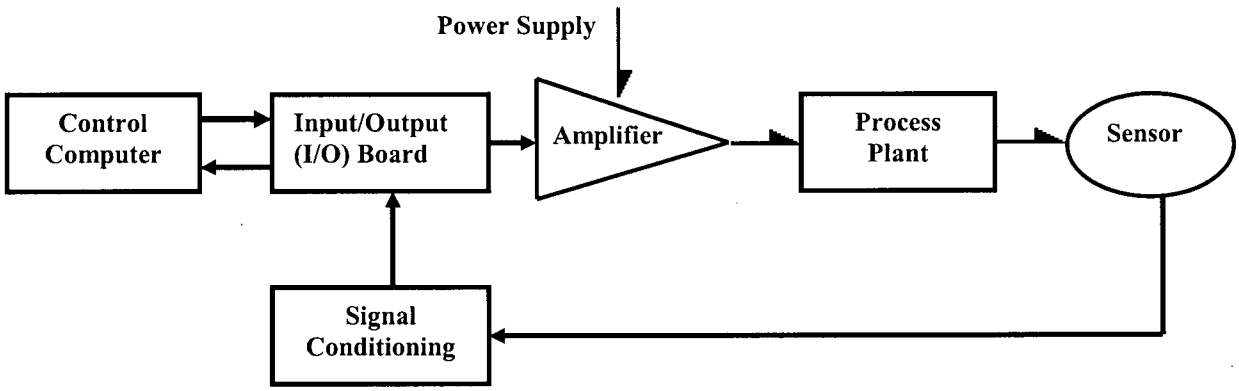


Figure 5.1. Schematic diagram of a typical mechatronic system.

These two domains can clearly be detected in the diagram of a mechatronic system, as shown in Figure 5.1. The high-power energy transfers are shown by half arrow, while the information transfers are shown by full arrow. The two domains are connected with each other through two basic components: sensor and power amplifier. By means of a sensor, an output variable (or a state variable) of the system is measured and fed into the information domain, while the power amplifier acts as a power source modulated by its input information signal.

Power and information interactions between components result in dynamic interactions between them (De Silva, 2003; De Silva, 2005). Due to the component interactions, generally, the entire system should be simulated and analyzed in an integrated manner. For this purpose, a common language is required that can effectively function in different engineering fields involved in a mechatronic system. Most commonly used modeling/simulation tools are appropriate only for a single specific domain. Hence they are not particularly appropriate for mechatronic systems. Bond graphs (BG) are proved to be an effective modeling method for mixed systems (Granda, 2002; Amerongen, 2003; Karnopp *et al.,* 2000). It has a unified graphical representation for lumped-parameter systems, which provides a common and core language for describing the basic elements and connections in different engineering fields that typically appear in a mechatronic system. In particular, it is a domain-independent tool for representing a mixed physical system. Figure 5.2 shows an example of a BG model. The basic elements and connections will be explained in succeeding sections.

BG is based on analyzing the flow of energy, which is the product of a *flow* variable (e.g., velocity, current, and flow rate) to an *effort* variable (e.g., force, voltage, and pressure), through the interconnections of components which are called *ports*. The final product of a bond graph

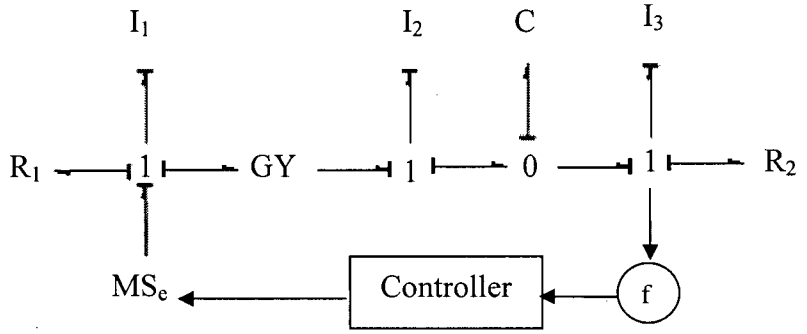


Figure 5.2: A simple BG model.

model is the state space dynamic equations of the system. The information domain of the system can be modeled by the block diagram representation and be connected to the power domain by a modulated power source, as shown in Figure 5.2.

BGs have three embedded strengths in design applications (Seo *et al.*, 2003):

- A wide range of systems can be modeled because of the multi- and inter-domain nature of BGs. Multi-domain systems (combined electrical, mechanical, hydraulic, pneumatic, and thermal systems) can be modeled using a common notation, which is especially important in the design of mechatronic systems.
- It provides high computational efficiency in the evaluation of design alternatives. This feature makes it suitable for use in design optimization.
- BG has graphical methodology to represent mixed systems. Any system model can be generated by a combination of bond and node components, rather than by the specification of equations.

The graphical representation of BGs provides an open architecture. This means that a free composition of bonds and nodes can be added to different locations of a model to create a new model with a new topology. In addition, any two system models can exchange a branch of their BG models to create two new models, which have some characters inherited from the initial models. This unbounded growth capability of BGs provides capability to explore a wide range of topologies in the process of design and optimization, especially by integration with evolutionary algorithms. In other words, the use of BGs for mechatronic design can result in an optimization tool, which is not restricted only to the sizing, but to optimize the topology of the model as well. This interesting feature of BGs has been utilized in the development of an evolutionary mechatronic tool in the present thesis, which is further illustrated in Chapter 6.

In this thesis a BG simulation tool is developed for use in design evaluation. A new formulation for the derivation of state space dynamic equations of a model is developed, which facilitates creation of a BG tool (Behbahani and de Silva, 2006c). This chapter presents the underlying process of the developed mechatronic simulation tool.

5.1 Bond Graph Terminology

Basically, bond graph modeling is based on the analysis of the flow of energy between components or elements. The connection of an element to another element is called a *port*, where there is power interaction between the particular element with the rest of the system

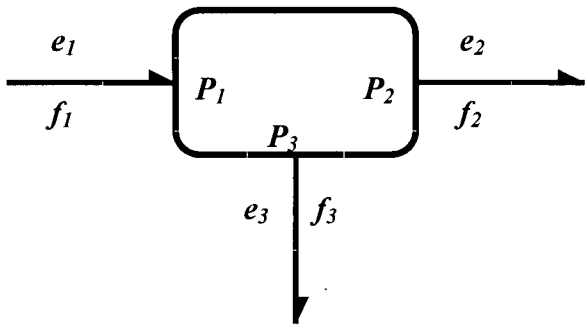


Figure 5.3: A multi-port element.

(Figure 5.3). The flow of energy between ports is shown by a line which is called *bond.* When two elements are connected, there are power interactions between them. The power can be in different forms; specifically, mechanical, electrical, hydraulic, or thermal. Power is always the product of two variables, called *power variables* (Table 5.1); in particular, mechanical power is the product of force and velocity while electrical power is the product of voltage and current. Bond graphs provide a common language to classify all power variables in different fields. In the BG terminology, all power variables are called either *effort* variables or *flow* variables, and are denoted by the symbols *e(t)* and *f(t)*, respectively. Force, torque, pressure and voltage are examples of effort variables whereas linear velocity, angular velocity, flow rate, and current are examples of flow variables. The flow of power between two connected elements through a port can then be expressed as the product of an effort variable and a flow variable.

$$P(t) = e(t).f(t) \tag{5.1}$$

There are two other important variables in describing a dynamic system. These variables are *momentum p(t)* and the *displacement q(t)* and are called *energy variables* (Table 5.1). Momentum and displacement are the time integrals of effort and flow, respectively.

Table 5.1. Power and energy variables in different domains.

Generalized Variable	Mechanical (Translation)	Mechanical (Rotational)	Electrical	Hydraulic
Effort, e	Force, F	Torque, τ	Voltage, v	Pressure, P
Flow, f	Velocity, v	Angular velocity, ω	Current, i	Flow rate, Q
Momentum, p	Linear momentum, p	Angular momentum, H	Flux linkage, λ	Pressure momentum, P
Displacement, q	Displacement, x	Rotation angle, θ	Charge, q	Volume, V

$$p(t) = \int e(t).dt$$
$$q(t) = \int f(t).dt \quad (5.2)$$

Then the energy flow can be described as:

$$E(t) = \int p(t).dq = \int q(t).dp \quad (5.3)$$

The direction of the power flow is shown by a half arrow on the bond. It is the direction of the power flow at any instant of time when the effort and flow both happen to be positive (or, both negative).

5.2 Basic Elements and Junctions

There are several primary 1-port element types (Karnopp *et al.*, 2000):

- *Resistance, R*: the 1-port resistance is an element in which there is a static relation between effort and flow variables, like electrical resistance, and mechanical damper (Figure 5.4).

$$e = R.f \quad (5.4)$$

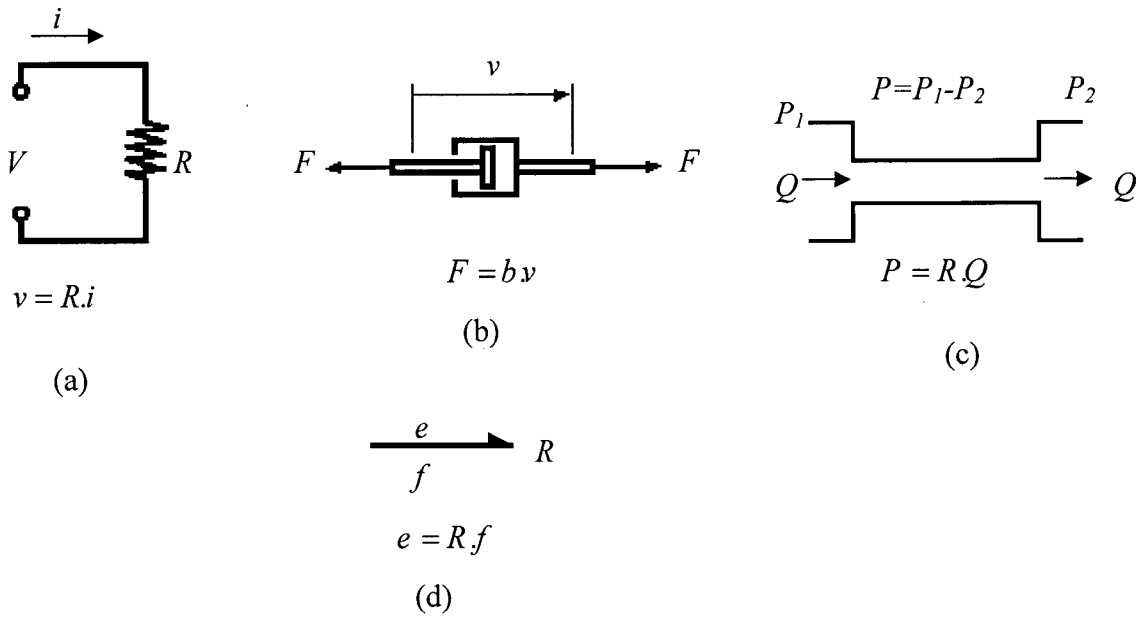


Figure 5.4: Resistor element. (a) Electrical resistance; (b)Mechanical damper; (c) Hydraulic connection with pressure drop; (d) General representation of resistor element in BG.

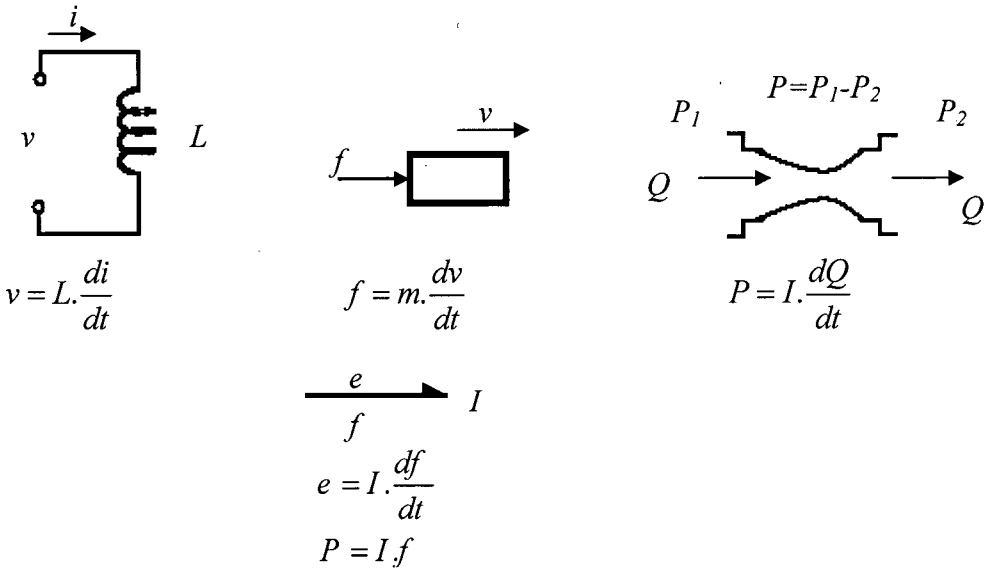


Figure 5.5: Inductance element; (a) Electrical inductance; (b) Mechanical inertia or mass; (c) Hydraulic nozzle; (d) General representation of an inductance element in BG.

- *Inductance, I*: the 1-port inductance is an element in which a constitutive law exists between the momentum p and the flow f. Examples are electrical inductance and mechanical inertia/mass (Figure 5.5).

$$p = I.f \tag{5.5}$$

- *Capacitance, C*: the 1-port capacitance is an element in which displacement q and effort e are related through a static constitutive law. Examples are electrical capacitor and mechanical spring (Figure 5.6).

$$q = C.e \tag{5.6}$$

- *Effort source, S_e*: a source of effort is an element that supplies energy to the system with a specified effort value; e.g., force generator, voltage generator, and pressure supply. The effort variable in the output port is impressed by this element. In contrast, this element accepts the flow variable impressed by the rest of the system to it.

- *Flow source, S_f*: a source of flow is an element that supplies energy to the system with a specified flow value; e.g., mechanical shaker, current generator, and constant

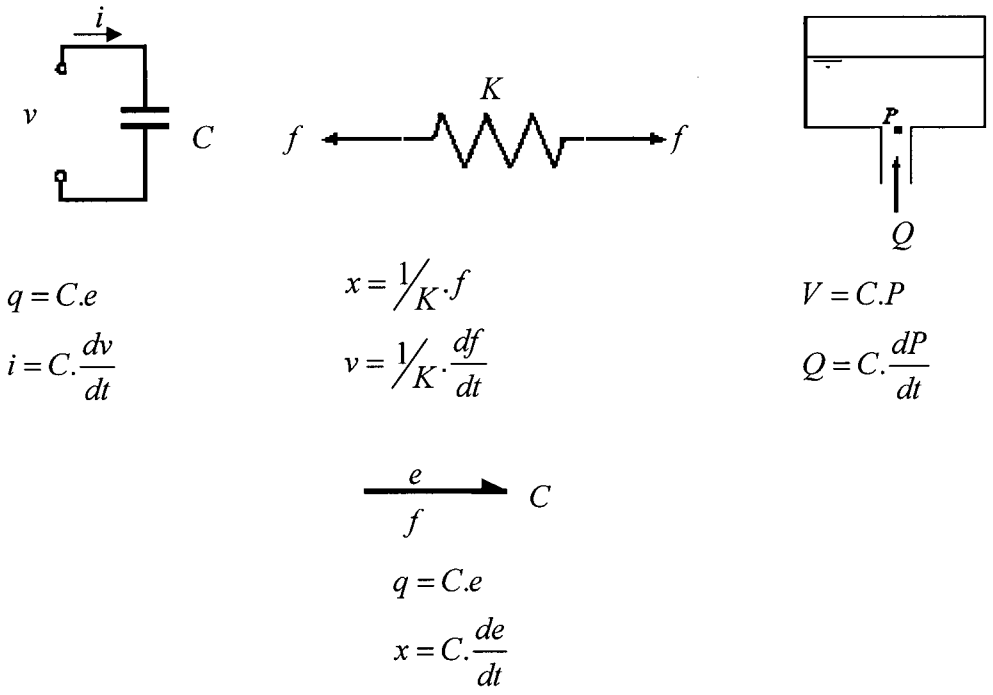


Figure 5.6: Capacitor elements: (a) Electrical capacitor; (b) Mechanical spring; (c) Hydraulic accumulator, or gravity tank; (d) General representation of a capacitor element in BG.

displacement pump. The flow variable in the output port is impressed by this element. In contrast, this element accepts the effort variable impressed by the rest of the system to it.

- *Modulated effort source, MS_e*: this is a special case of effort source where its value is specified by an input information signal. It is useful to model the interconnection between the power and information domains of a mechatronic system.

- *Modulated flow source, MS_f*: similar to an effort source, MS_f is a flow source whose value is specified by an input information signal.

- *Effort sensor*: it is an element that measures the effort variable in a port and generates an information signal.

- *Flow sensor*: it is an element that measures the flow variable in a port and generates an information signal.

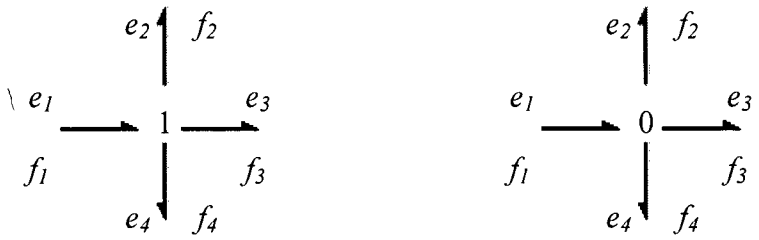


Figure 5.7: Multi-port junctions: (a) 1-junction; (b) 0-junction.

Also there are two types of junctions:

- *1-junction*: in this junction, all connected bonds have a common flow variable, and their effort variables add to zero (compatibility). For the 1-junction shown in figure 5.7(a), the following equations can be written:

$$\begin{aligned} f_1 &= f_2 = f_3 = f_4 \\ e_1 &= e_2 + e_3 + e_4 \end{aligned} \tag{5.7}$$

- *0-junction*: it represents a junction where all connected bonds have a common effort variable, and their flow variables add to zero (continuity). For the 0-junction shown in figure 5.7(b), the following equations can be written:

$$\begin{aligned} e_1 &= e_2 = e_3 = e_4 \\ f_1 &= f_2 + f_3 + f_4 \end{aligned} \tag{5.8}$$

Furthermore, there are two types of 2-port elements: the *transformer TF* and *gyrator GY* elements. Figure 5.8(a) shows the representation of a transformer in BGs. If this transformer is an ideal transformer, its constitutive laws are a follows:

$$\begin{aligned} e_1 &= r_T e_2 \\ r_T f_1 &= f_2 \end{aligned} \tag{5.9}$$

where r_T is called *transformer modulus*. Transformer can be used in the modeling of rigid levers, gear pairs, electrical transformers, and hydraulic rams.

(a) (b)

Figure 5.8: Two port elements: (a) The transformer; (b) The Gyrator.

Figure 5.8(b) shows the representation of a gyrator in BGs. The constitutive laws of the gyrator are:

$$\begin{aligned} e_1 &= r_G f_2 \\ r_G f_1 &= e_2 \end{aligned} \tag{5.10}$$

where r_G is *gyrator modulus*.

5.3 Modeling of Mechatronic Systems

Bond graph modeling of a mechatronic system can be viewed as an object oriented modeling (OOM) method, even though BG was developed a long time by Professor Henry M. Paynter at Massachusetts Institute of Technology in 1959 (Paynter, 1961), long before the concept of object-oriented modeling was coined. Basically, the concept of OOM means that different subsystems of a mixed machine can be modeled separately and be connected with each other to create the main model. Furthermore, subsystem models can be reused again in modeling of other machines.

Any real physical machine has a hierarchical structure. The system may be composed of some lower level subsystems. Each subsystem may also consist of some lower level subsystems. In object-oriented bond graph modeling, the model of each subsystem can be considered as an *object*. Consequently, a general model for components that commonly appear in mechatronic systems can be established and reused wherever it is necessary. Each subsystem model (say object) has several inherent parameters; i.e., values of the parameters in the sub-model. In order to be able to reuse the model of a component as an object, its model needs to be generated so that it receives all its inherent parameters from outside, through the so-called *interfaces*. Whenever the model is used as an object, its inherent parameters should be provided by the higher level sub-model (which can be the main model) which has used that particular object.

The following procedure can be used for modeling a mixed system:

- Properly shrink the system to several smaller parts and construct a hierarchical pattern for it.
- If you have access to the model of a component in the lowest level, find the required inherent parameters for it and simply substitute it into your model. Otherwise construct a bond graph model for that particular component. It is recommended to generate the model of the component as an object; i.e., it receives all its inherent data through interfaces. By this, if a similar component exists in other parts of the machine, you can reuse the created object just by defining its actual parameters. In addition, you can add it to a so-called library of objects for modeling other machines.
- Create the model of the higher level subsystem or the main model by interconnecting the sub-models and proper definition of their inherent parameters.

5.3.1. Modeling of Mechanical Systems

For systems of mechanical nature, use the following procedure (Karnopp *et al.*, 2000):

- Detect all the mechanical nodes; i.e., nodes with distinct velocity, and find the mass/inertia at the node.
- Consider a 1-junction for each mechanical node. Attach the mass/inertia element directly to its corresponding 1-junction.
- Any force generating element, mechanical capacitor (spring), or mechanical resistor (damper) in the system will appear between two mechanical nodes. If there is such an element between two nodes, insert a 0-junction between the corresponding 1-junctions and join that element to it (Figure 5.9a). If there are more than one element between two nodes, first bundle them together by connecting them to a 1-junction, and then connect the unit to the 0-junction that you have inserted between corresponding nodes (Figure 5.9b).
- Assign a power direction to all bonds
- If one of the 1-junctions represents the zero-velocity reference point, delete that 1-junction and its corresponding bond.

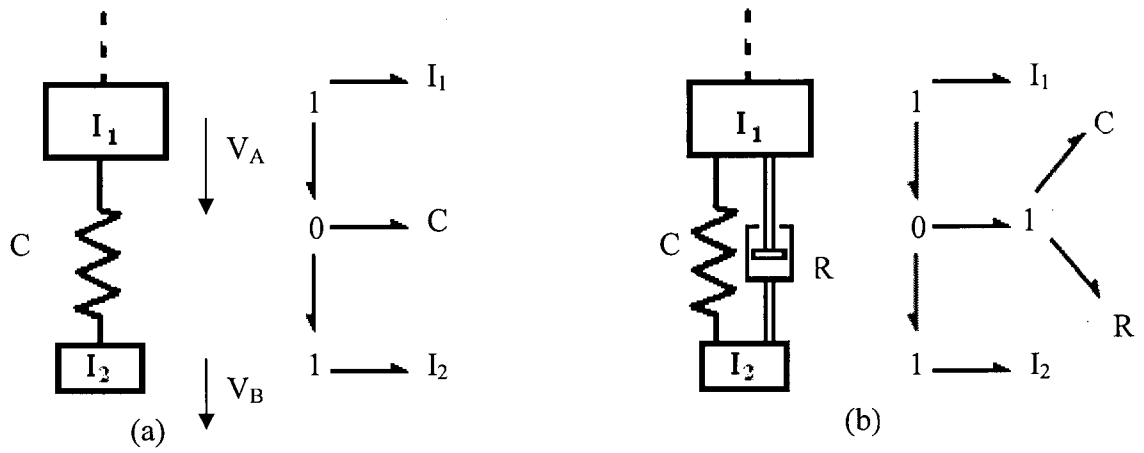


Figure 5.9: Insertion of mechanical elements between distinct nodes:
(a) Single element; (b) Multiple elements.

5.3.2. Modeling of Electrical Systems

To model an electrical circuit, use the following procedure (Karnopp *et al.*, 2000):

- Detect all the nodes in the circuit which have a distinct voltage. Consider a 0-junction for each of these nodes.
- If there is any 1-port element (resistor, inductance, capacitor, or source element) between two nodes of the circuit, insert a 1-junction between the corresponding 0-junctions and join that element to it (Figure 5.10a). If there are more than 1 element between two nodes, first bundle them together by connecting them to a 0-junction, and then connect resulting unit to the 1-junction that you have inserted between corresponding nodes (Figure 5.10b).

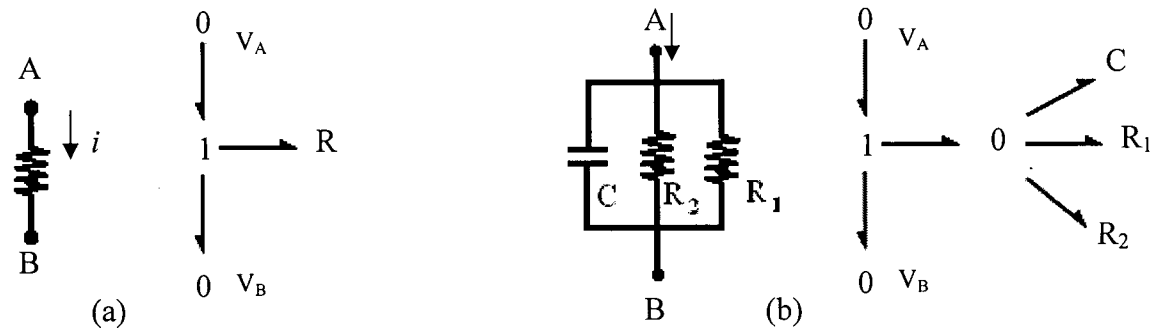


Figure 5.10: Insertion of electrical elements between distinct nodes:
(a) Single element; (b) Multiple elements.

- Assign a power direction to all bonds.
- If there is an explicit ground in the circuit, delete its corresponding 0-junction and its bond; otherwise, chose an arbitrary ground and delete its corresponding 0-junction and bond.

5.3.3. Modeling of Hydraulic Systems

Modeling of hydraulic systems is very similar to the procedure of modeling of electrical circuits. The following procedure may be used for hydraulic systems (Karnopp *et al.*, 2000):

- Consider a 0-junction for any point in the hydraulic circuit that has distinct pressure.
- If there is a 1-port hydraulic component between two distinct nodes, insert a 1-junction between the corresponding 0-junctions and join the element to it (similar to Figure 5.10a). If there are more than 1 element between two nodes, first bundle them together by connecting them to a 0-junction, and then connect the resulting unit to the 1-junction that you have inserted between corresponding nodes (similar to Figure 5.10b).
- Assign a power direction to all bonds.
- Delete the 0-junction corresponding to the reference pressure and its bond.

5.3.4 Simplification of a Bond Graph Model

When a bond graph model has been created, we may be able to simplify it by the following set of rules:

1-Eliminating redundant junctions: if a junction is the only link between two other junctions or between a junction and a 1-port element, that junction can be eliminated (Figure 5.11).

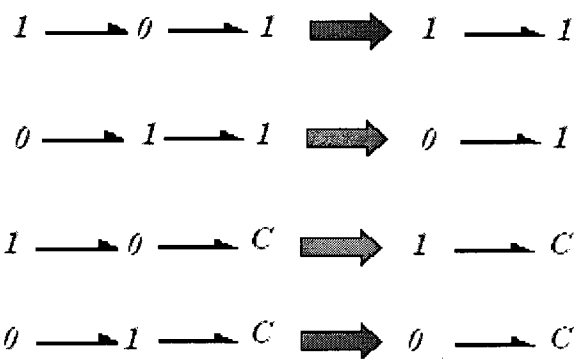


Figure 5.11: The conditions that a junction is redundant and can be eliminated.

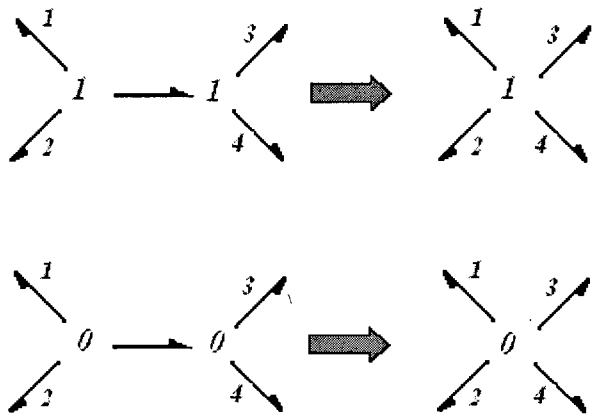


Figure 5.12: The conditions that two junctions can be melted together.

2-Melt equal junctions: if two or more similar junctions are connected together, they can be merged together (Figure 5.12).

If there is a transformer between two junctions with the same type, then they can be melted and elements on one of them can be transferred to the other, but a transformation has to be applied on them. If the transformer connects node n to node m and flow direction is from n to m, then if elements on node m are transferred to node n, their equalized value will be (Figure 5.13):

$$
\begin{aligned}
R_{eq} &= R \cdot r_T^{\,2} \\
I_{eq} &= I \cdot r_T^{\,2} \\
C_{eq} &= C / r_T^{\,2} \\
S_{e_{eq}} &= S_e \cdot r_T \\
S_{f_{eq}} &= S_f / r_T
\end{aligned}
\qquad (5.11)
$$

where r_T is the module (ratio) of the transformer. If node n is melted to m, then inverse of the r_T should be substituted into the above equations.

If there is a gyrator between two junctions of dissimilar type, then they can be melted. The transformation of the elements of one of them to the other is more complex than in the case of a transformer. In addition to the change in the magnitude of the elements, their nature also changes. A capacitor appears as an inductance and vice versa. A source of effort becomes a source of flow and vice versa.

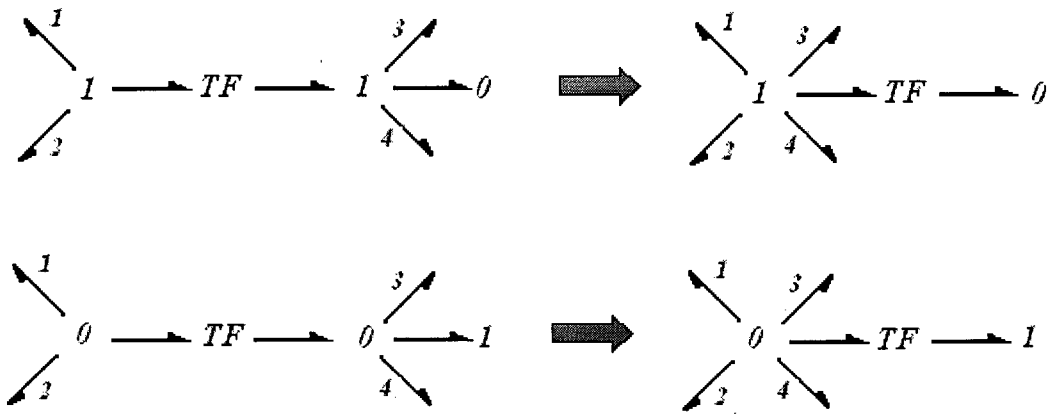


Figure 5.13: Melting of equal junctions connected by a transformer.

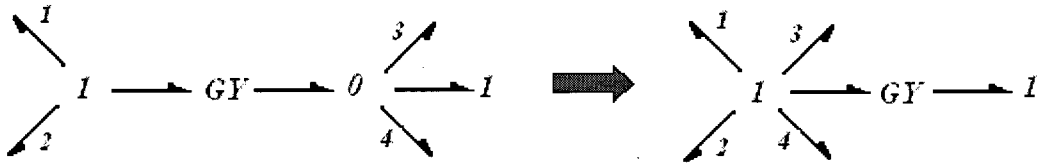


Figure 5.14: Melting of equal junctions connected by a gyrator.

If a gyrator connects node *n* to *m* and flow direction is from *n* to *m*, then if elements on node *m* are transferred to node *n*, their equalized value will be (Figure 5.14):

$$
\begin{aligned}
R_{eq} &= r_G^2 / R \\
I_{eq} &= C \cdot r_G^2 \\
C_{eq} &= I / r_G^2 \\
S_{e_{eq}} &= S_f \cdot r_G \\
S_{f_{eq}} &= S_e / r_G
\end{aligned}
\qquad (5.12)
$$

where r_G is the module of the gyrator. If node *n* is melted to *m*, then inverse of the r_G should be substituted into the above equations.

3-Melting of similar elements connected to a junction: a 0-junction is like a parallel connection of electrical circuits. Therefore, if there are similar elements connected to a 0-junction, following rules are used:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots$$
$$\frac{1}{I_{eq}} = \frac{1}{I_1} + \frac{1}{I_2} + \ldots \qquad (5.13)$$
$$C_{eq} = C_1 + C_2 + \ldots$$

A 1-junction is like a series connection in electrical circuits. Therefore, if there are similar elements connected to a 1-junction, following rules are used:

$$R_{eq} = R_1 + R_2 + \ldots$$
$$I_{eq} = I_1 + I_2 + \ldots \qquad (5.14)$$
$$\frac{1}{C_{eq}} = \frac{1}{C_1} + \frac{1}{C_2} + \ldots$$

5.3.5 Causality Analysis

Both flow and effort variables exist at a port and only one of them can be controlled. One of them will be the input to the port and the other one will be the output from that port. In other words, when two components are connected together by a bond the effort variable in the bond causes one of the elements to respond with a flow, while this flow causes the other element to respond with an effort. In bond graphs, the direction of the effort signal is shown by a short perpendicular line at the end of a bond which is called a *causality stroke.*

Causality analysis is a very important process in bond graph modeling because of several reasons. First, it is required in the derivation of the state space equations. In each junction, one and only one of the connected bonds has dominant causality. For example, all connected bonds to a 0-junction have similar effort variable. Therefore, only one of the bonds connected to a 0-junction can have the effort as its output and all the other bonds have to have opposite causality. It will be explained later that in writing the equations of each junction, it is necessary to know which connected bond has dominant causality.

Second, causality analysis checks whether the model is feasible or not. Any causality conflict in a model implies that model is not physically feasible. This bond graph simulation tool will be integrated with a genetic programming search tool, which is described in Chapter 6. Genetic programming is known as a random search tool. As a result there exists the possibility of generating a large number of unfeasible models. It follows that causality analysis is useful in making the modeling process time efficient, by avoiding the analysis of models that are not feasible.

Table 5.2 (Karnopp *et al.*, 2000) shows the possible causal forms and causal relations for basic 1-port elements. Source elements have a restricted causal form, because they impress either effort or flow output, regardless of the system connected to them. In storage elements (capacitor and inertia), the integral causality is preferred.

Tables 5.3 and 5.4 (Karnopp *et al.*, 2000) show the possible causal forms for 2-port elements and multi-port junctions. In a 0-junction, it is important that only one connected port has an effort output, and others accept the associated effort as an input and impress flow outputs. 1-junctions have opposite causality. The port which has dominant causality specifies the flow value of all the remaining connected ports. Other ports accept this flow variable as an input, and impress effort outputs.

Table 5.2: Possible causality forms for 1-port elements and associated equations.

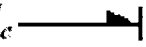
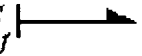
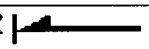
Element	Causal Form	Causal Relation
Effort Source		$e(t) = E(t)$
Flow Source		$f(t) = F(t)$
Resistor		$e = R.f$
		$f = e/R$
Capacitor		$e = \frac{1}{C}\int_{i}^{t} f.dt = q/C$
		$f = \frac{d}{dt}(C.e)$
Inertia		$f = \frac{1}{I}\int_{i}^{t} e.dt = p/I$
		$e = \frac{d}{dt}(I.f)$

Table 5.3: Possible causality forms for 2-port elements and associated equations.

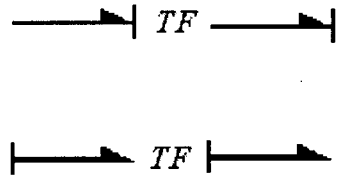
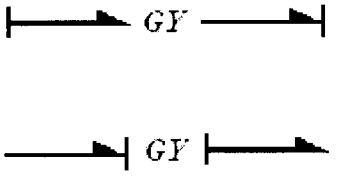
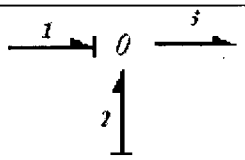
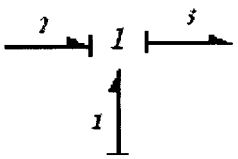
Element	Causal Form	Causal relations
Transformer		$f_1 = f_2/m$, $e_2 = e_1/m$ $e_1 = m.e_2$, $f_2 = m.f_1$
Gyrator		$e_1 = r.f_2$, $e_2 = r.f_1$ $f_1 = e_2/r$, $f_2 = e_1/r$

Table 5.4: Possible causality forms for multi-port junctions and associated equations.

0-junction		$e_2 = e_3 = e_1$, Bond 1 is dominant $f_1 = -(f_2 + f_3)$
1-junction		$f_2 = f_3 = f_1$, Bond 1 is dominant $e_1 = -(e_2 + e_3)$

The underlying process for causality analysis of a BG model is as follows (Karnopp *et al.*, 2000):

1. Source elements have a restricted causality form. Consider a source element. Assign its required causality. Extend causal implications as far as possible, using the causality requirement of 2-port and multi-port elements.
2. Repeat step 1 for all sources. If any causal conflict happens in this stage, it can be implied that the model is not physically feasible.

3. Consider a storage element whose causality is not specified in previous stages. Assign its preferred causality. Extend causal implications as far as possible, using the causality requirement of 2-port and multi-port elements.
4. Repeat step 3 for all unassigned sources.
5. If the causality of the model is not complete yet, chose any unassigned R-element. Assign an arbitrary causality to it. Extend causal implications as far as possible, using the causality requirement of 2-port and multi-port elements.
6. Repeat step 5 for all unassigned R elements.
7. If the causality of the model is still not complete, chose an unassigned bond and assign arbitrary causality to it. Extend causal implications as far as possible, using the causality requirement of 2-port and multi-port elements.
8. Step 7 is repeated for all unassigned bonds.

If any conflict happens during the causality analysis process, the model is unacceptable, and therefore makes dynamic simulation unnecessary.

5.3.6 Incorporation of the Information Domain

Components of a mechatronic system are linked with each other through flow of energy and/or flow of information. Controller part of a mechatronic system has usually very low level of energy transfer, which is negligible compared to the energy flow between other parts. However, the information flow associated with this low level energy flow is not negligible. Hence, a mechatronic system can be divided into two domains: power domain and information domain. The bond graph modeling considers the flow of energy between components and derives state space dynamic equations of the system. In other words, it accounts for the high-energy part of the system. The resultant dynamic equations from a bond graph tool can be linked to SIMULINK in order to incorporate the control and information domains of the system into the power domain and simulate the equations. The other benefit of this is that numerical subroutines of SIMULINK can be used to simulate the state space equations.

5.4 Derivation of State Space Equations

Bond graph modeling leads to straightforward derivation of the state space dynamic equations of a system. These equations can be solved/simulated in the time domain by using numerical

methods. In this section, a new bond graph formulation is described, which facilitates the creation of a bond graph program. It is based on the matrix representation of the governing equations in local nodes and then creation of global state space equations by assembling them. This is somewhat similar to and inspired by the finite element representation of equations. A bond graph simulation tool is developed in this thesis in order to handle the required simulations. The process of obtaining the state space equations using bond graphs is accomplished in several stages, including:

1-Local matrix formation

2-Assembly

3-Final reduction of the matrices

5.4.1 Local Matrix Formation

The first step is to analyze the power flow for each junction and derive the local equations for each junction. Energy exchange from a junction can be considered in two parts:

1-Energy exchange to the basic elements connected to the junction.

2-Energy exchange to other junctions connected to it.

In each junction, one connected bond has dominant causality; either a bond connected to a 1-port element or a bond connected to another junction. As mentioned, all bonds connected to a 1-junction have a common flow variable and all bonds connected to a 0-junction have a common effort variable. The bond that has the dominant causality is the bond which specifies this common magnitude in the junction, and it has to be flowed through all the other bonds. Effort and flow variables in the dominant bond are power variables which are used to express the governing equations.

Three local equations are written for each junction:

1- Compatibility/continuity equations. For a 0-junction, flow variables add to zero (continuity equation), and for a 1-junction effort variables add to zero (compatibility equation).

2- The equation that expresses the common variable in the junction (i.e., flow variable in 1-junction, or effort variable in 0-junction). It is actually the constitutive law of the element that specifies the common variable in the junction. If an interconnecting bond has dominant causality, no equation is written in this stage.

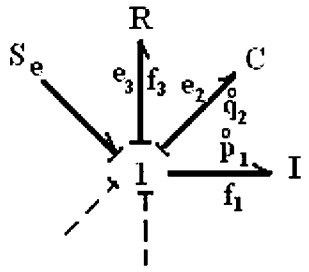


Figure 5.15: A typical 1-junction.

3- The equation describing the variation of the co-energy variable in the junction.

Consider a 1-junction. The most common case is that a bond connected to an inductor has dominant causality in the junction (Figure 5.15). In this case, the momentum in this bond is one of the state variables of the system. If a capacitor is also connected to the junction, the displacement in the bond connected to this capacitor is also a state variable of the system. If no capacitor is connected, the third equation is left empty. The equations in the junction are as follows:

$$\begin{aligned} \dot{p}_1 &= e_1 = -e_2 - e_3 + S_e + \sum e_{bonds} = -Rf - \frac{q_2}{C} + S_e + \sum e_{bonds} \\ f &= \frac{p_1}{I} \\ \dot{q}_2 &= f \end{aligned} \tag{5.15}$$

or

$$\begin{Bmatrix} \dot{p}_1 \\ f \\ \dot{q}_2 \end{Bmatrix} = \begin{bmatrix} 0 & -R & -1/C \\ 1/I & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{Bmatrix} p_1 \\ f \\ q_2 \end{Bmatrix} + \begin{Bmatrix} S_e \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} \sum e_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.16}$$

The energy exchange with other junctions will be taken into account in the assembly step which comes next. If it is ignored temporarily, the equations for this typical 1-joint can be written in the following form:

$$\begin{Bmatrix} \dot{p}_1 \\ f \\ \dot{q}_2 \end{Bmatrix} = K_{mat} \begin{Bmatrix} p_1 \\ f \\ q_2 \end{Bmatrix} + F_{mat} \tag{5.17}$$

The matrix K_{mat} can be called a *local stiffness matrix* and vector F_{mat} can be viewed as a *local force vector*. It should be mentioned that equation 5.17 is still not valid. It will be valid after bringing the effect of the other bonds into account.

This is the case when an I-element has dominant causality and has assigned the flow variable. If no I-element exists, then an R-element may assign the flow. In this case:

$$\begin{Bmatrix} e_1 \\ f \\ \dot{q}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & -1/C \\ 1/R & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} e_1 \\ f \\ q_2 \end{Bmatrix} + \begin{Bmatrix} S_e \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma e_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.18}$$

If the flow is assigned by a source of flow, then:

$$\begin{Bmatrix} e_1 \\ f \\ \dot{q}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & -1/C \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} e_1 \\ f \\ q_2 \end{Bmatrix} + \begin{Bmatrix} S_e \\ S_f \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma e_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.19}$$

Finally, if it is assigned by an interconnecting bond, the second equation is left empty at this stage:

$$\begin{Bmatrix} e_1 \\ f \\ \dot{q}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & -1/C \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} e_1 \\ f \\ q_2 \end{Bmatrix} + \begin{Bmatrix} S_e \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma e_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.20}$$

Now consider a typical 0-junction and suppose that a bond connected to a capacitor has dominant causality in the junction (Figure 5.16). The displacement in this bond is one of the state variables of the system. The momentum in the bond connected to the inductor is another state variable of the system (if exist). Writing equations for this junction yields:

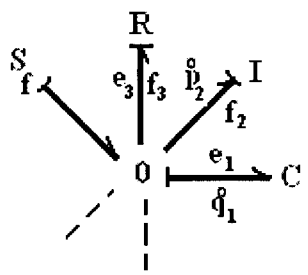


Figure 5.16: A typical 0-junction.

$$\dot{q}_1 = f_1 = -f_2 - f_3 + S_f + \Sigma f_{bonds} = -\frac{e}{R} - \frac{p_2}{I} + S_f + \Sigma f_{bonds}$$

$$e = \frac{q_1}{C} \tag{5.21}$$

$$\dot{p}_2 = e$$

or

$$\begin{Bmatrix} \dot{q}_1 \\ e \\ \dot{p}_2 \end{Bmatrix} = \begin{bmatrix} 0 & -1/R & -1/I \\ 1/C & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} q_1 \\ e \\ p_2 \end{Bmatrix} + \begin{Bmatrix} S_f \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma f_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.22}$$

The energy exchange is ignored temporarily and will be taken into account in the assembly step, which comes next. The equations can then be written in the following form:

$$\begin{Bmatrix} \dot{q}_1 \\ e \\ \dot{p}_2 \end{Bmatrix} = K_{mat} . \begin{Bmatrix} q_1 \\ e \\ p_2 \end{Bmatrix} + F_{mat} \tag{5.23}$$

The matrix K is analogous to a local stiffness matrix and vector F is analogous to a local force vector. As the energy exchange has been ignored temporarily, equation (5.23) is still not valid. It will be valid after completing the assembly step which accounts for interconnecting bonds.

This is the case when a C-element is assigned the effort variable. In the absence of a capacitor, an R-element may assign the flow. In this case:

$$\begin{Bmatrix} f_1 \\ e \\ \dot{p}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & -1/I \\ R & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} f_1 \\ e \\ p_2 \end{Bmatrix} + \begin{Bmatrix} S_f \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma f_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.24}$$

If the effort is assigned by a source of effort (not usual), then:

$$\begin{Bmatrix} f_1 \\ e \\ \dot{p}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & -1/I \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} f_1 \\ e \\ p_2 \end{Bmatrix} + \begin{Bmatrix} S_f \\ S_e \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma f_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.25}$$

Finally, if it is assigned by a bond, one has:

$$\begin{Bmatrix} f_1 \\ e \\ \dot{p}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 & -1/I \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} . \begin{Bmatrix} f_1 \\ e \\ p_2 \end{Bmatrix} + \begin{Bmatrix} S_f \\ 0 \\ 0 \end{Bmatrix} + \begin{Bmatrix} \Sigma f_{bonds} \\ 0 \\ 0 \end{Bmatrix} \tag{5.26}$$

5.4.2 Assembly

Now, each junction has created a 3×3 local stiffness matrix and a 3×1 local force vector. These local matrices are assembled in a global stiffness matrix in their proper places. For example, first to third rows and columns are allocated for the first joint. The stiffness matrix of the second joint is placed between 4^{th} and 6^{th} rows and columns. Its force vector is also placed between 4^{th} and 6^{th} rows.

Next step is to take into account the energy flow through the connected joints and assemble all local equations into a global set of equations. It should be mentioned that after the simplification step (explained in section 5.3.4) all connected junctions are in different form. Two conditions may apply for this process:

a) If the associated bond is not dominant for causality of the connected joints (Figure 5.17a), the following items should be added to the global stiffness matrix. If for example, the associated bond connects node *n* to node *m* and if the flow of the energy is from node *n* to node *m*, then:

- The second energy variable of node *m* should be subtracted from the first equation of node *n*.
- The second energy variable of node *n* should be added to the first equation of node *m*. It means:

$$K_{3n-2,3m-1}=1$$
$$K_{3m-2,3n-1}=-1 \qquad (5.27)$$

b) If the associated bond is dominant for the causality of the connected joints (Figure 5.17b), then the following items are added to the stiffness matrix:

(a) (b)

Figure 5.17: Connection of junctions: (a) Connecting bond is not causality dominant; (b) Connecting bond is causality dominant.

- The second energy variable of node n is equal to the first energy variable of node m.
- The second energy variable of node m is equal to negative of the first energy variable of node n.

$$K_{3n-1,3m-2} = 1$$
$$K_{3m-1,3n-2} = -1 \tag{5.28}$$

Four conditions may arise if two junctions are connected together through a transformer (Figure 5.18). After the simplification step, transformers can only be placed between dissimilar junctions.

If the associated bonds are not dominant for causality of the connected joints, then:

a) If the flow of energy is from *1*-junction to *0*-junction (Figure 5.18a):

$$K_{3n-2,3m-1} = r_T$$
$$K_{3m-2,3n-1} = -r_T \tag{5.29}$$

b) If the flow of energy is from *0*-junction to *1*-junction (Figure 5.18b):

$$K_{3n-2,3m-1} = 1/r_T$$
$$K_{3m-2,3n-1} = -1/r_T \tag{5.30}$$

If the associated bonds are dominant for the causality of the connected joints, then:

c) If the flow of energy is from *1*-junction to *0*-junction (Figure 5.18c):

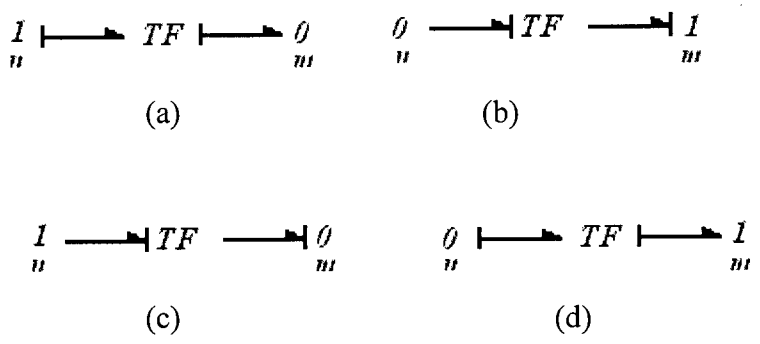


Figure 5.18: Assembly conditions for a transformer.

$$K_{3n-1,3m-2} = 1/r_T$$
$$K_{3m-1,3n-2} = -1/r_T \tag{5.31}$$

d) If the flow of energy is from *0*-junction to *1*-junction (Figure 5.18d):

$$K_{3n-1,3m-2} = r_T$$
$$K_{3m-1,3n-2} = -r_T \tag{5.32}$$

Similar four conditions may occur when two junctions are connected together through a gyrator (Figure 5.19). After the simplification step, a gyrator can only be placed between similar junctions.

If the associated bonds are not dominant for causality of the connected joints, then:

a) If two 1-junctions are connected (Figure 5.19a):

$$K_{3n-2,3m-1} = r_G$$
$$K_{3m-2,3n-1} = -r_G \tag{5.33}$$

b) If two 0-junctions are connected (Figure 5.19b):

$$K_{3n-2,3m-1} = 1/r_G$$
$$K_{3m-2,3n-1} = -1/r_G \tag{5.34}$$

If the associated bonds are dominant for the causality of the connected joints, then:

c) If two 1-junctions are connected (Figure 5.19c), one has:

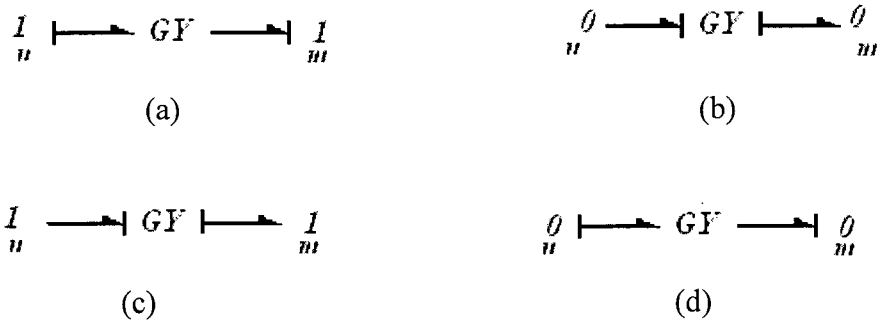


Figure 5.19: Assembly conditions for a gyrator.

$$K_{3n-1,3m-2} = 1/r_G$$
$$K_{3m-1,3n-2} = -1/r_G \tag{5.35}$$

d) If two 0-junctions are connected (Figure 5.19d):

$$K_{3n-1,3m-2} = r_G$$
$$K_{3m-1,3n-2} = -r_G \tag{5.36}$$

5.4.3 Final Reductions

After performing the assembly step, redundant variables can be eliminated form the equations. Only storage elements in each junction will provide a state variable for the system. Other variables appear in the algebraic form and can be eliminated simply by algebraic manipulations. After this step, only the state space variables will remain.

5.5 Summary

In view of the presence of components from different engineering domains in a mechatronic system, integrated design of these systems requires a domain-free simulation tool. This chapter presented the development of a bond graph simulation tool. Bond graphs represent a lumped-parameter, domain-free method which provides a core language to represent components in different domains in a unified modeling environment. State space dynamic equations of the system are derived through bond-graph modeling by analyzing flow of energy between components.

In this chapter a new matrix-based formulation and procedure were presented to facilitate the development of a bond graph simulation tool. It is somewhat similar and inspired by the finite-element method. Once the bond graph model is created, it may be possible to simplify it by using several simple rules. The modeling process is preceded only if the model is feasible and can pass the causality analysis. The main objective of creating a bond graph simulation tool in this work is to integrate it with a genetic programming optimization tool, resulting in an evolutionary mechatronic tool. Causality analysis can prevent analysis of many infeasible models which are generated randomly by genetic programming. Next, for each junction in the model, local matrices are derived. These local matrices are assembled into global matrices to account for energy transfer between junctions. The state-space dynamic equations of the system are finally derived by eliminating redundant equations in the developed global matrices. The

bond-graph simulation tool, which is developed in the present work, has the potential to be integrated with SIMULINK in order to integrate the model of the information part of the system to the power part.

Chapter 6

Integration of Bond Graphs and Genetic Programming

Design, modeling, control, and analysis of machines are typical activities of an engineer. For a mechatronic system, these activities tend to be rather complex because of the presence of components from different domains and dynamic interactions between them. Approaches for these activities for a mechatronic system should somewhat deviate from those for conventional, non-mechatronic systems, in part due to the need for integrated design and analysis in mechatronic systems. Multi-domain system design tends to be somewhat complex and this is further exacerbated by the fact that most available simulation tools and design standards are developed for a single domain, and may not be appropriate in the design of a multi-domain system.

Due to its multi-domain nature and associated dynamic interactions, the design of a real mechatronic system should take an integrated and concurrent approach rather than a traditional and sequential approach where subsystems are designed separately in their own domains while neglecting the dynamic interactions with the rest of the system. Due to dynamic interactions between subsystems of a system, the final design of the system may not be optimal if the subsystems are optimized separately. In particular, the optimization process should be performed on the entire system and the decisions involved in the system design, including not just the parameter values for sizing of the components, but also the topological arrangement of the components which affects dynamic interactions.

In a typical mechatronic problem, the objective is to find the optimal structure of a machine that best satisfies the user-defined requirements of the solution. This entails finding the best topology and size of the machine. The topology is defined as the gross number of components, their type, and the way they are interconnected together, while sizing refers to finding the numerical parameter value of the elements (Koza *et al.,* 2000). If the topology of the best solution was known, a variety of optimization tools could be used to find the corresponding size. The mechatronic problem is usually complex because the topology of the best solution is not necessarily obvious. Since the topology of a system has naturally an unlimited growth capability and consists of unlimited conditions, finding the topology of a mechatronic system opens a huge search space for the designer. Due to this complexity, topology design is usually viewed as a problem which needs human intelligence and should be performed by experts.

The main challenge in creating a tool for autonomous design and modeling of mechatronic systems is to develop a method to effectively explore all possible topologies for a mechatronic system. This requires two main characteristics:

1- Since the topology can grow unlimitedly, the exploration method needs to have a topologically open-ended structure. Genetic Programming (GP) is a branch of evolutionary algorithms which has open-ended growth possibility of solutions.

2- Generally, a mechatronic system contains components from different engineering fields; mechanical, electrical, hydraulic, and so on. Although these fields are physically different, their basic components and governing equations are analogical. To facilitate topology exploration, a core language is necessary to express components from different domains in a unified language. Bond graph modeling (BG) is a domain-free graphical representation of multi-domain systems. All components from different domains can be modeled in a unified environment. This also meets the requirement of integrated design for mechatronic systems.

There exists research work on integration of bond graph modeling and genetic programming for the development of a unified mechatronic design tool with the aim to synthesize the topology and size of a mechatronic system (Seo *et al.*, 2003; Fan *et al.*, 2004). They have been inspired by earlier work on the application of genetic programming for the automated design of electrical circuits (Koza *et al.*, 1999; Koza *et al.*, 2000). They provide several construction functions and terminals for the creation of linear bond graph models. Genetic programming has been used to explore different topologies with the aim to optimize a goal function. The developed tool has been successfully used in the automated design of several mixed systems. Subsequently, Wang *et al.*, (2005) has extended the same approach to knowledge interaction in mechatronic system design. To unify the control system design with mechanical system design, they applied the concept of "controller design in physical domain." In this method, a controller is represented by an equivalent physical system; hence, the controller can also be represented by the bond graph method.

This chapter extends the earlier work to nonlinear problems, and applies the resulting nonlinear methodology to automated system identification (Behbahani and de Silva, 2005b, c, 2006c). Several new construction functions are added to the previous work in order to account for possible nonlinearities that may exist in a mechatronic system.

The developed software tool is applicable to any mechatronic problem provided that a fitness evaluation scheme is established for the application of genetic programming in the particular

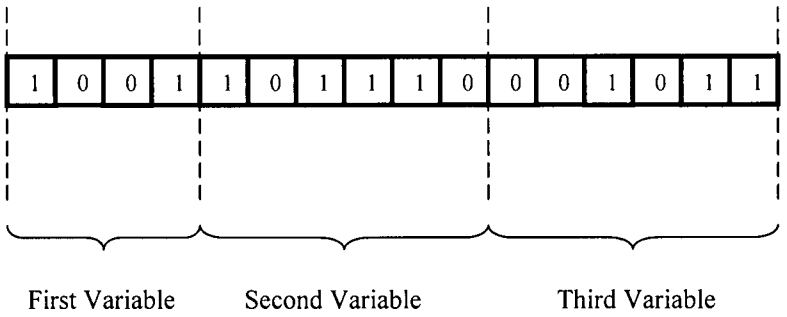
problem. In evaluating the capabilities of the developed tool, the problem of system identification of a nonlinear system is used here as an illustrative example. Most common system identification techniques are applicable for linear systems, and also they do not generate details of the physical structure of the model of the system. In proper system identification, one needs to determine both topology and size of the system, which can be a difficult task due to the possible complexity of a mixed system. The chapter investigates relevant issues of this problem.

6.1 Genetic Programming

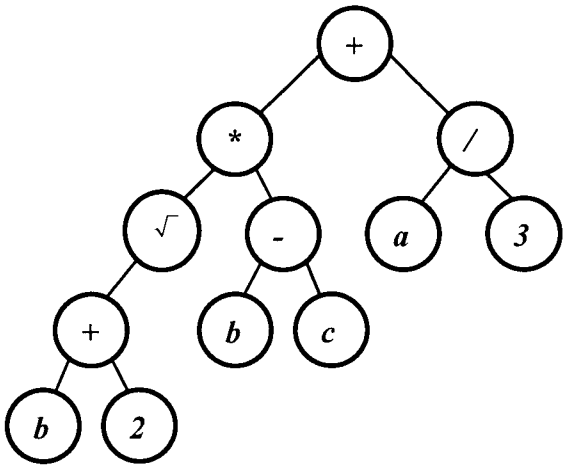
Evolutionary algorithms and in particular genetic algorithms are optimization methodologies inspired by the evolution of biological species in nature. They are stochastic search methods based on Darwin's principle of survival of the fittest. Genetic programming (GP) is a branch of evolutionary computing as is genetic algorithms (GA). The main difference between genetic programming and conventional genetic algorithms comes from the particular representation of the solution (see GP tutorial).

Before starting to apply genetic algorithms on a particular problem, a methodology has to be developed to represent the possible trial solutions for the particular problem in a chromosome-like structure. In conventional GA, each individual (trial) solution is represented as a series of zeros and ones, whereas in GP a tree formed by construction functions represents a solution (Figure 6.1). This latter method of representation was initially introduced in order to represent mathematical equations in a tree-like format. An example is shown in Figure 6.1. This tree-like representation makes genetic programming suitable for concurrent sizing and topology optimization. Each tree results in a solution alternative and is considered a chromosome, where its branches are considered genes. As clear from Figure 6.1, a change in the genes of a chromosome can result in a new topology, in addition to a new size of the result.

GP imitates evolution of species in the nature to evolve an embryo solution toward a solution with good fitness. It is based on Darwin's principle of the survival of the fittest. Basically, members of a generation that have higher fitness will have better chance to survive or to be selected to participate in reproduction. In addition to high reproduction opportunity which the best solutions have, they also have a greater chance to be copied in the next population without any change. This ensures the convergence of the GA process. Since the best solution in each generation will be better or at least the same as the best in the previous generation, the process of evolution will probabilistically converge to the best solution.



(a)



(b)

Figure 6.1: Chromosome-like representation of individual solutions:
(a) Conventional GA; (b) Tree-like representation in GP.

A key purpose of system identification of a complex system is to determine the topology and size that can best represent the behavior of the system. The capability of genetic programming for topology and sizing optimization has been a motivation to apply it for system identification in this work.

6.1.1 Genetic Programming Terminology

Several basic concepts in genetic programming are described now.

6.1.1.1 Embryo

Two main concepts in this methodology are the embryo model and the construction functions. Embryo includes very basic specifications of solution and remains unchanged in all alternative

solutions. Embryo needs to have several modifiable sites. These are places where the embryo can be extended through them.

6.1.1.2 Construction Functions

Construction functions are the alphabet of creation of new models. They are functions, which are applied to the embryo in order to create new and extended models. They have open architecture. This means, they can create new modifiable sites in the model. This feature makes them grow like branches of a tree. Proper termination functions are also necessary to prevent unlimited growth of branches. Each tree-like combination of construction functions represents a trial solution for the problem at hand. In fact, a mapping between each optional solution and a combination of defined construction functions should be established.

6.1.1.3 Fitness Function

The most difficult and also most important concept of genetic programming is the fitness function. The destiny of each trial solution depends on its fitness. The best and the fittest solutions have a greater chance for survival in the new generation and also for participation in a larger number of reproduction operations. On the other hand, solutions with a lower level of fitness are more likely to die without producing any offspring in the new generation. The fitness function is a function reflecting the degree of satisfaction of each trial solution. It should vary greatly from one type of solution to the next and should be such that the farther a model is from the ideal solution, the more it will be penalized.

6.1.1.4 Reproduction Operations

A new generation of trial solutions is created from previous generation by reproduction operations, including

- Crossover
- Mutation
- Survival (selection)

The most important of these, which has the highest percentage of reproduction probability is the crossover operation. In the crossover operation, two solutions are mated to form two new solutions or offspring. The parents are randomly selected from the population, but in such a way that the members with higher fitness have a greater chance to be selected. The creation of the

offspring from the crossover operation is accomplished by exchanging a gene branch between parents (Figure 6.2). An important improvement of genetic programming over conventional genetic algorithms is its ability to accomplish crossover from the same solution (see GP tutorial). In Figure 6.3 the same parent is used twice to create two new children.

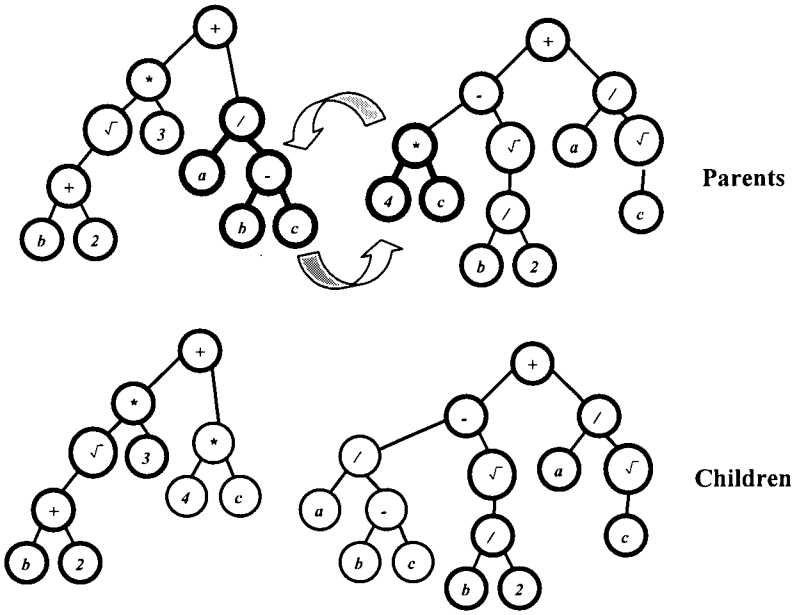


Figure 6.2: Crossover genetic operation on a tree-like GP representation.

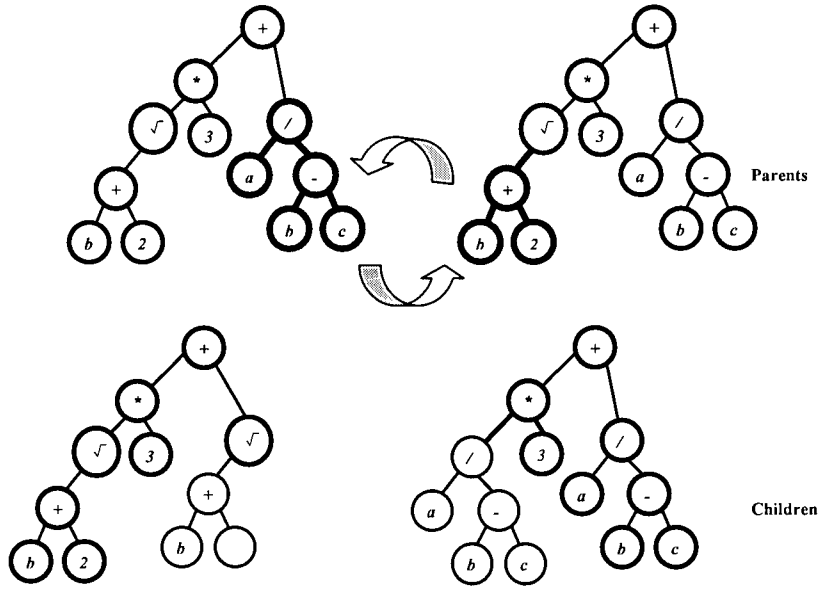


Figure 6.3: Crossover reproduction operation between identical parents (GP tutorial).

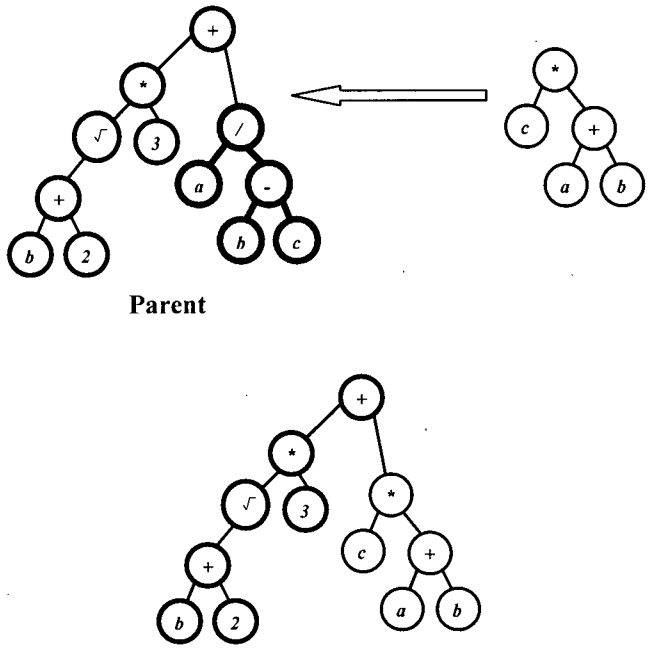


Figure 6.4: Mutation genetic operation in a tree-like representation of GP (GP tutorial).

Mutation is another operation of genetic programming. In mutation, an individual is selected randomly, but in such a way that the solutions with higher fitness have a greater chance to be selected. In this operation a gene branch is selected randomly and replaced by a new randomly created gene branch, as shown in Figure 6.4. Mutation gives a very important strength to genetic programming. Hill climbing is the most important challenge of optimization techniques. In this process, the optimization path may get locked into a local optimum and be unable to escape that. The mutation operation of genetic programming avoids or at least reduces the local minimum problem. It works like a gun-shot on the optimization process.

In addition to the above-mentioned operations, there is a critical operation where the best solutions are always copied to the next generation in order to guarantee that the best solutions are retained. This operation also guarantees that no degradation will happen in the best solutions of the population, and in turn guarantees the convergence of the optimization process.

6.1.2 Flowchart of Genetic Programming

In order to effectively solve a problem by genetic programming, the following three items should be integrated:

1)A simulation tool

2)Fitness evaluation

3)Construction functions and terminals to establish a mapping between GP structures and each solution option.

To solve a problem by GP, first an *embryo* solution should be provided. The embryo reflects the very basic specifications of the solution, such as inputs, and outputs. It has some *modifiable sites* and some *open sites* for modification and extension of the embryo. *Construction functions* provide possible modifications and extensions available for different types of modifiable sites. By applying different random combinations of construction functions to the embryo, the first population of possible solutions is created. However, this first population will hardly include good solutions, yet different solutions can be ranked in terms of their fitness. Next generation of solutions is formed by genetic programming operations which are based on the Darwin's principle of survival of the fittest. Selection of the parents for a GP operation and the place in the chromosome for applying the particular operation are all performed with some level of randomness to mimic the natural evolution. Because of Darwin's principle, the next generation is likely to be better than the current generation, and eventually an optimal solution will be achieved by continuing this process. Another benefit of genetic programming is that mutation avoids or reduces undesirable effect of *hill climbing* where locking into a local maximum would be possible (see Introduction to genetic algorithms).

Genetic programming uses the following five steps to solve a problem (Figure 6.5):

1. Provide an embryo solution. Embryo solution includes very basic specifications of solution and remains unchanged in all alternative solutions. Embryo needs to have several modifiable sites. These are places were the model can be extended through them.

2. Generate an initial population of random solutions by the composition of some construction functions to an embryo solution. Being random, these initial solutions are not good.

3. Simulate each solution in the population and assign it a fitness value according to how well it solves the problem.

4. Create a new population of solutions by imitation of evolution of species. New generation is formed based on Darwin's principle of survival of the fittest through the following genetic operations:

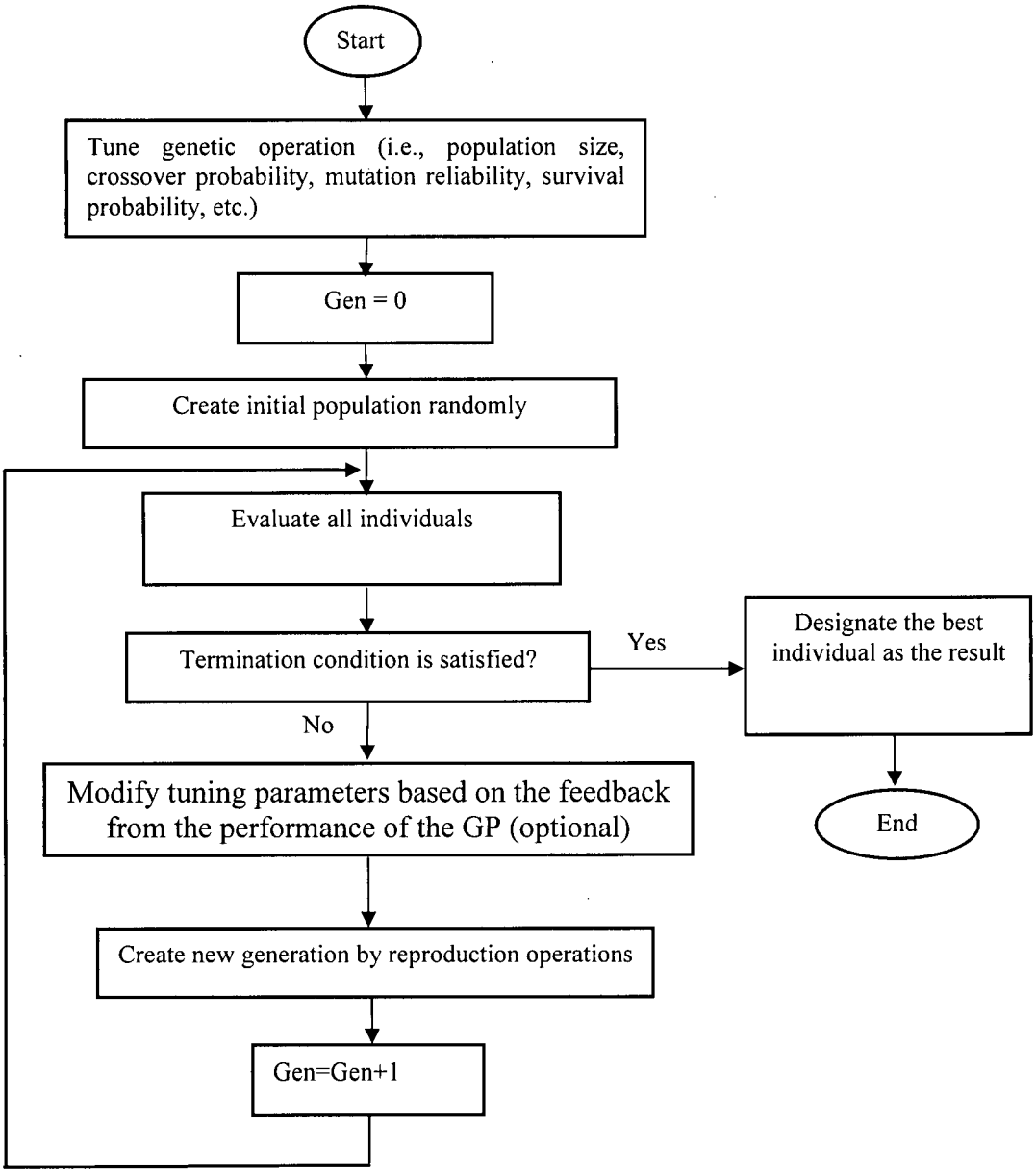


Figure 6.5: Flowchart of genetic programming.

- Copy the best existing solutions
- Create new solutions by mutation.
- Create new solutions by crossover.

5. Return to step 3 until satisfaction condition is met or a specified upper limit on the number of iterations is reached. The best solution that is resulted in this manner in any generation (the best-so-far solution) is designated as the outcome of genetic programming.

6.1.3 Selection Method

The essence of genetic programming is the selection of good solutions from a trial set of solutions, and making them participate in some genetic operations. Three methods have been used in literature for the selection of solutions having good fitness values (Karray and de Silva, 2004):

1. Probability method
2. Tournament method
3. Ranking method

The first method specifies selection probability for each individual solution based on its fitness. If $f\left(S_i\right)$ is the fitness of the solution S_i and $\sum_{j=1}^{N} f\left(S_j\right)$ is the total sum of fitness of all the members of the population, then the probability that the solution S_i will be selected is:

$$P(S_i) = \frac{f\left(S_i\right)}{\sum_{j=1}^{N} f\left(S_j\right)} \tag{6.1}$$

Another method for selecting the solution to be copied is tournament selection. Typically, the genetic program chooses two solutions randomly, and the solution with the higher fitness will win. This method simulates biological mating patterns in which two members of the same sex compete to mate with a third one of a different sex.

In the third method, the selection is done by the rank (not the numerical value) of fitness of the solutions of the population. In the present work, this method is used with some modification. Once a generated set of solutions is evaluated, they are ranked according to their fitness in the ascending order. Two parent solutions are necessary to perform the crossover operation. Two random numbers are created by this operation and then two parents are selected based on these random numbers. To give a greater opportunity to the solutions with high fitness, a mapping function is employed such that the random numbers, ranging between 0 and 1, are mapped mostly to high ranked members. In this thesis, the following mapping function is employed:

$$\frac{n}{N} = r^{\xi} \tag{6.2}$$

where, r is a random number, N is the total population, n is the rank of the selected solution, and ξ is a discrimination factor which is less than one. This mapping is shown in Figure 6.6. The

lower the value of ξ, the higher the probability of death of a weak member (i.e., higher discrimination). It is useful to tune this number in the GP process.

Each solution in the form of a GP tree is presented as a "chromosome." Very long chromosomes have been generated in the present work. A change in the genes of a long chromosome may not push it toward the optimal solution in an effective manner. A long chromosome may eventually attain a rather high fitness and create many more long chromosomes in the next generation. These chromosomes may lie at the top of the generation and prevent further improvement of the solution, or at least reduce the efficiency of the GP. In view of this, it is useful to introduce a penalty for having a long gene. However, the members that are ranked very high are made immune to this penalty.

The age of a chromosome is another factor that needs to be considered. Specifically, a penalty is introduced for the age of a chromosome. If a chromosome is quite old and yet it is not in a member of very high rank, it is desirable to delete it from the competition and let the younger solutions continue in the competition. However, members with very high ranks are kept immune to this penalty.

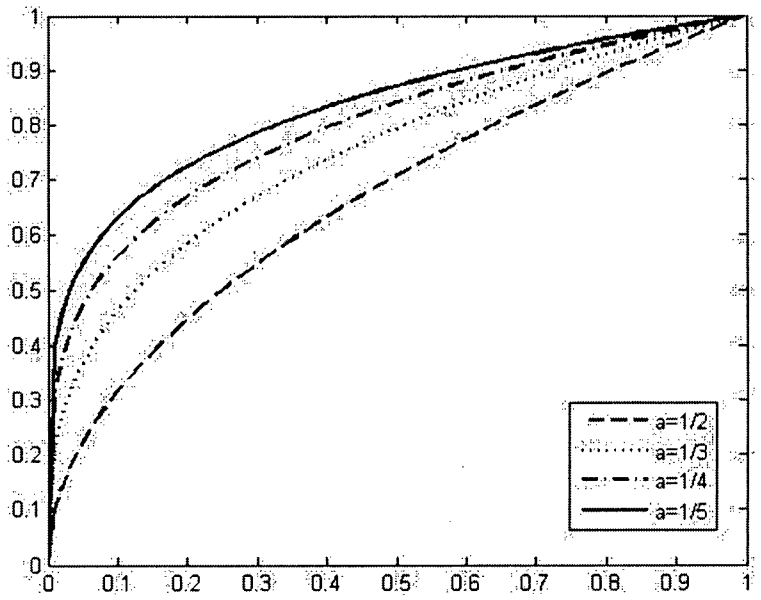


Figure 6.6: The mapping used to give higher chance of selection to better solutions.

6.2 Integration of Bond Graph Modeling and Genetic Programming

Modeling of mixed systems requires a common language that can effectively function in different engineering fields involved in a mechatronic system. Bond graphs (BG) are proved to be an effective and accepted modeling method for such systems. They provide a common and core language for describing basic elements and connections across different fields.

Representation of the mixed systems by BGs provides an embedded strength for BGs to be used in design applications (Seo *et al.,* 2003). Graphical representation of BGs allows their generation by the combination of bond and node components. This means any system model can be represented by a combination of bond and node components. This graphical representation of BGs also has an open architecture. This means that a free composition of bonds and nodes can be added to different locations of a model to create a new model with a new topology. This unbounded growth capability of BGs provides capability to explore a wide range of topologies in the process of design and optimization. In other words, the use of BGs in mechatronic design can result in an optimization tool, which will not be restricted to the system sizing aspect, but will optimize the topology of the model as well. This graphical flexibility of BG and the extensive search ability of GP have provided the rationale for integration of these two powerful tools.

The development of a bond graph simulation tool is presented in Chapter 5. A tool has been developed in the present work which is able to derive state space equations of a bond graph model. These equations are solved/simulated in the MATLAB environment, by using its powerful solvers.

6.2.1 Bond Graph Embryo Model

To solve any problem using integration of BGs and genetic programming, an embryo solution should be provided by the user. This bond graph embryo model should contain the fundamental information about the solution, such as inputs, and outputs. The embryo model is the common part of all the trial solutions which will be created in the process of optimization by genetic programming. It is problem dependent, and it is the user who should decide the level of the details in the embryo. As a general rule, the inputs and outputs of the problem which are involved in the fitness evaluation should be available in the embryo model.

In the embryo model, several places should be specified as modifiable sites. Considering the typical BG embryo model shown in Figure 6.7, three types of modifiable sites can be observed:

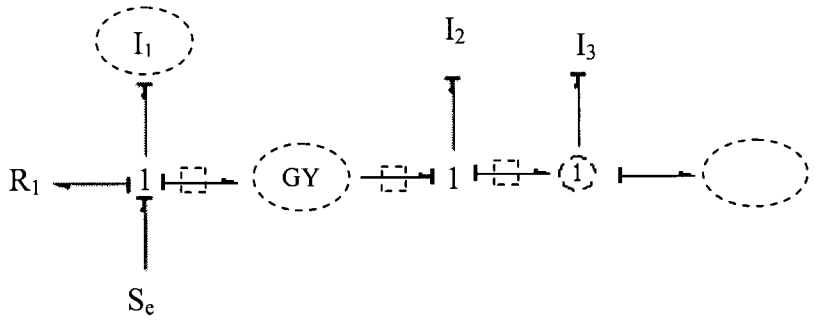


Figure 6.7: A sample BG embryo model.

1) Modifiable joints, shown by dashed circle
2) Modifiable bonds, shown by dashed square
3) Arithmetic sites, shown by dashed elliptic.

Modifiable joints allow for extension of the embryo by adding new elements to nodes. A basic one-port element or a useful compound object can be added to a node. In the present work, an element can be either linear or nonlinear.

Modifiable bonds allow extension of the connections between nodes. Arithmetic sites correspond to modifiable numerical values of the elements in the embryo and also to elements that are added to it. Arithmetic sites are necessary for the sizing optimization of a created model. In the present work, if an element is linear, a constant number is created by the arithmetic functions to represent the physical magnitude of the element. For a nonlinear element, an evolutionary function-generation scheme has been developed to find the best function that represents the element.

6.2.2 Construction Functions

Next necessary item is the establishment of construction functions and terminals to provide the mapping between a GP tree-like representation method and all possible bond graph models.

Each of the already-explained modifiable sites has its own possible construction functions. In the present work, the construction functions developed in an earlier work by Seo *et al.,* (2003) are used, and several new construction functions are developed as well to account for possible nonlinearities in the system. All construction functions have an open architecture. It means, they introduce new modifiable sites and also keep old modifiable sites for further improvement. This feature causes the individual chromosomes to grow like branches of a tree. It also makes

necessary to consider an *End Function* for each type of modifiable site to prevent unlimited growth.

Six types of functions are developed in the present work, for integration of GP and BG:

1)Insert junction: Inserts a junction (either 1 or 0 junction)

2)Add element: adds one of the basic 1-port elements

3)Arithmetic: performs arithmetic operations for finding the value of an element

4)Compound: a useful combination of nodes and bonds is added

5)Add-friction function

6)Add-backlash function

Next these functions are illustrated.

6.2.2.1 Add-Element Functions

Consider a modifiable node. A basic 1-port element (i.e., resistor, capacitor, or inductor) may be added to this node. Figure 6.8 shows a modifiable node before and after adding a 1-port element, a resistor in this case. When the new element is added, first of all, the old modifiable site should be kept modifiable for further modifications. Second, the added element needs a numeric value, either by specifying its value if it is linear, or by providing an arithmetic function. This opens a new modifiable arithmetic site in the model. In addition, the bond which connects the element to the initial node is also a site where the model can be further extended. It

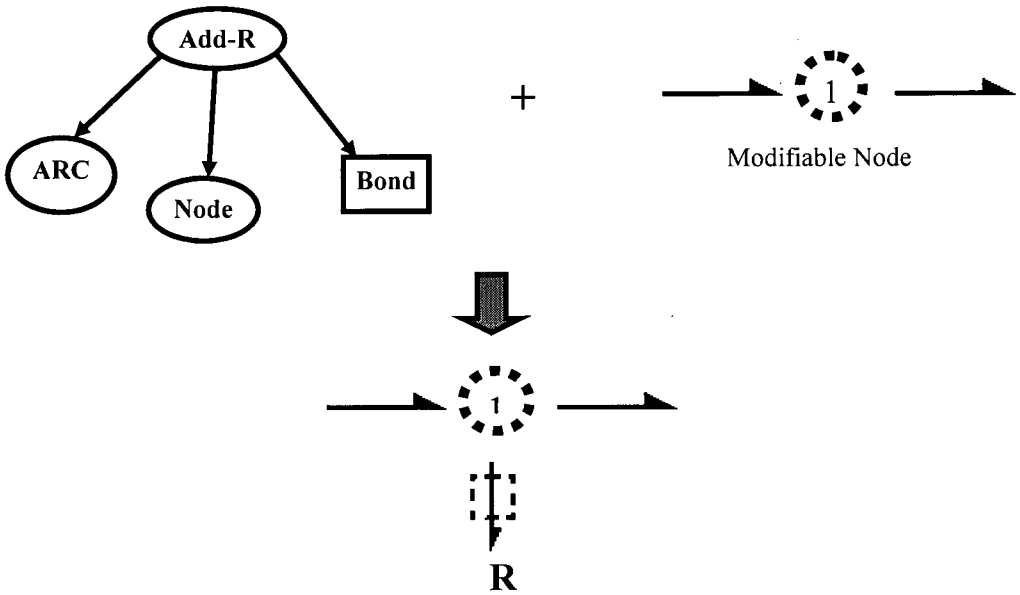


Figure 6.8: Add-element construction function.

means that when an add-element function is applied to a modifiable mode, two new modifiable sites are created and the old modifiable site is also kept for further modification (Figure 6.8). Consequently, this function has three arguments. The first one indicates the value of the added element with arithmetic functions, the second one is the old modifiable site for further modification, and the last one is a bond-type site for inserting a junction at the connection point of the added bond. There are three types of Add-Element functions: *Add-R, Add-C*, and *Add-I*. As these functions may give rise to an unending sequence of add operations, it is necessary to have the *End-Node* function to terminate the operation.

6.2.2.2 Insert-Junction Functions

Consider again a modifiable bond. A junction may be inserted between bonds. This function has three arguments or creates 3 new modifiable sites. Two of them are bond-type and the third one is node-type, and they are shown in Figure 6.9.

As these functions may result in an open sequence of operations, it is necessary to provide a termination operation for them. Accordingly, for each modifiable bond, three options are available: *Insert-0, Insert-1*, and *End-Bond.*

6.2.2.3 Arithmetic Functions for Linear Elements

Each added new element or a modifiable element in the embryo model needs a value. If the element is linear, its magnitude can be represented by a constant number. Otherwise it has to be

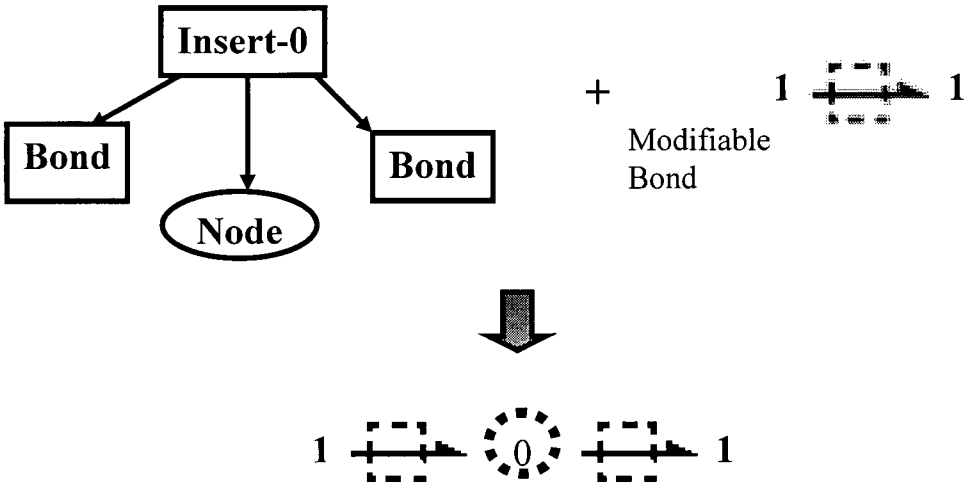


Figure 6.9: Insert-junction construction function.

represented as a function of other variables in the system. In the linear case, each arithmetic modifiable site may be a random number or a combination of arithmetic functions and random numbers. Two kinds of arithmetic functions are used: addition and subtraction. Each function has two arguments, which may also be either random numbers or a set of arithmetic functions. Regardless, the result of an arithmetic sub-tree will be a number, which represents the numerical value of the element. The resulting number may be interpreted in various ways, depending on the designer. For example, it may be interpreted as the logarithmic value of an element (Seo *et al.*, 2003). In the present thesis, it is interpreted as the change in an initial default value. The value of a parameter ζ is calculated by:

$$\zeta = \zeta_d(1+\rho.\varepsilon) \tag{6.3}$$

Here ε is the result of the arithmetic sub-tree, ρ is a modification factor, and ζ_d is the default value for parameter ζ. It is found desirable to apply a rather high ρ in the beginning of a GP operation for coarse modification and then reduce it later for fine modification.

6.2.2.4 Arithmetic Functions for Nonlinear Elements

In the present thesis, basic one port elements are not restricted to be linear. The value of a nonlinear element is a function of other parameters in the system. In particular, it can depend on the state space variables of the node to which the particular element is connected. Here, several construction functions are developed to create different functions representing the value of a nonlinear element. These construction functions have been used in the genetic programming optimization technique, to explore different functions and find the optimum one for representing the value of a particular nonlinear element. If the nonlinear element is in the embryo model, the user can determine how many variables may appear in the variation of the value of the element. If the nonlinear element is created in the process of optimization, then it is considered to depend only on the state variables of the node connected to the particular nonlinear element (Figure 6.10).

Table 6.1 gives a list of arithmetic operators used in the present work and the number of arguments needed for each of them. Each argument can be a function itself, a random number, or one of the variables of the function. Figure 6.11 shows an example of creation of a function by using these construction functions.

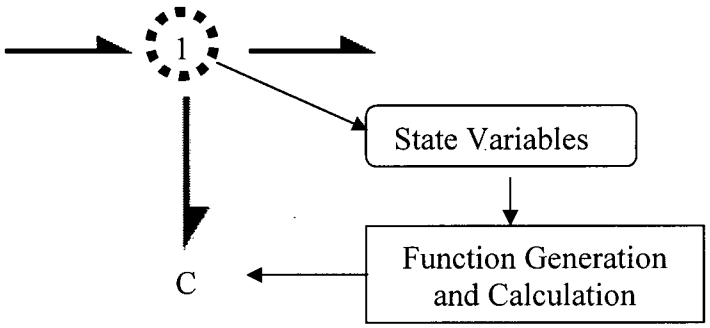
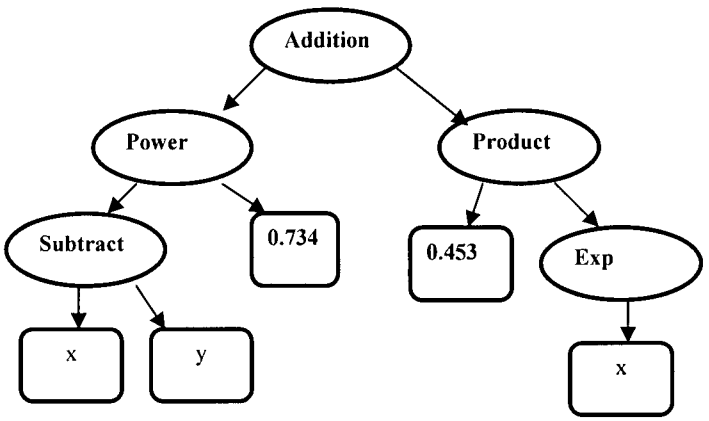


Figure 6.10: Function generation for nonlinear elements.

Table 6.1: Arithmetic operators and associated number of arguments.

Operation	Number of arguments
Addition	2
Subtraction	2
Product	2
Division	2
Power	2
Sign	1
Exp	1
Log	1



$$f = (x-y)^{0.734} + 0.453\exp(x)$$

Figure 6.11: An example of function generation.

6.2.2.5 Compound Object-Creation Functions

It is useful to have functions for creating compound objects, which may be commonly needed in the modification operations. Although it is possible that these compound elements are created through the normal operation of a GP by means of the basic functions which have been developed, it is more effective to establish compound functions to facilitate creation of these objects. For example, one useful compound object which may be commonly needed in building the mechanical model of a machine is a Mechanical Degree of Freedom (MDOF). The BG model of an MDOF is shown in Figure 6.12. It adds one mechanical degree of freedom to a mechanical system, which corresponds to adding two state space variables. It may be added to a modifiable node; consequently, a new function that is available for a modifiable node is Add-MDOF. It has 6 arguments, which are shown in Figure 6.12.

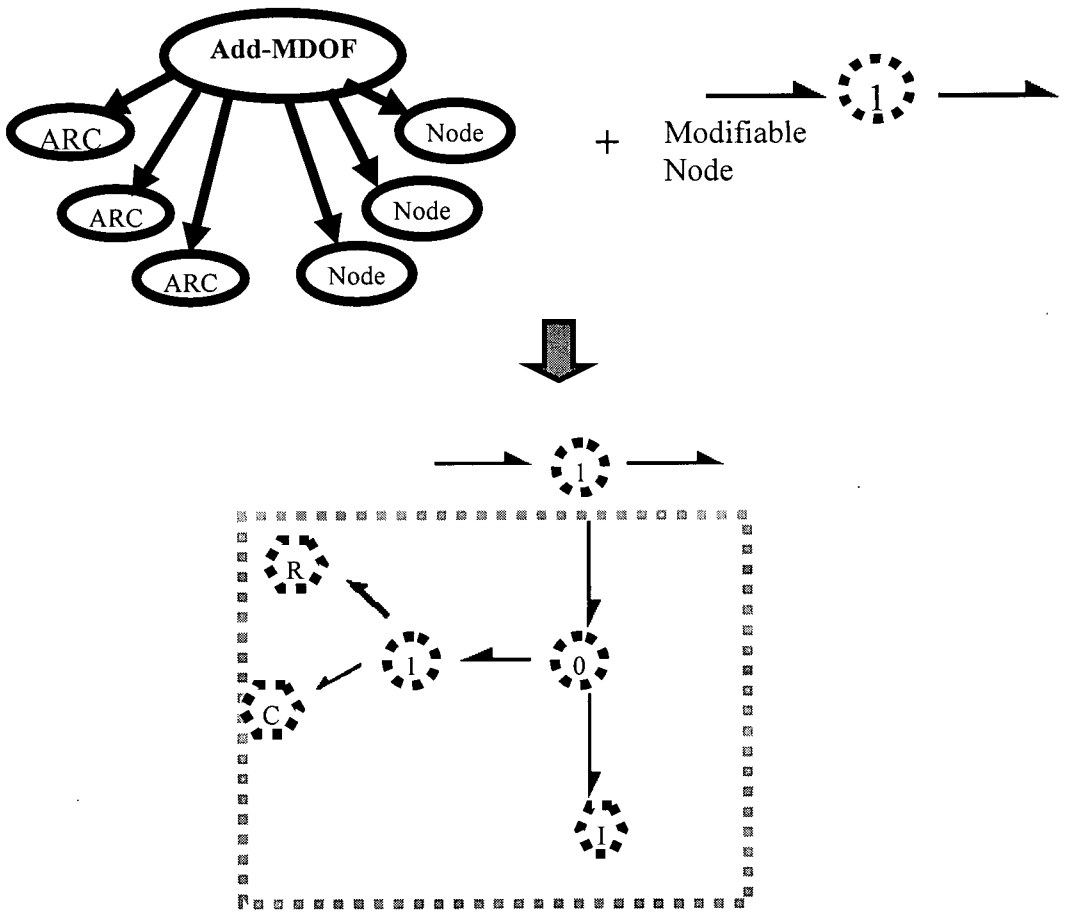


Figure 6.12: Add mechanical degree of freedom (MDOF) construction function.

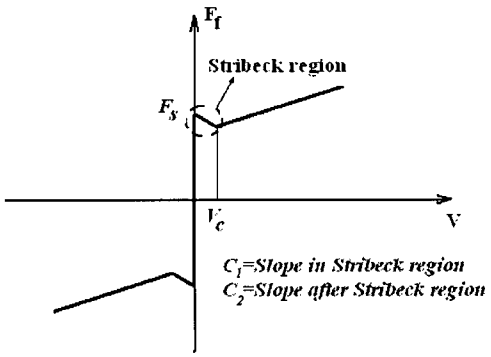


Figure 6.13: Stribeck friction model.

6.2.2.6 Add Friction Function

Friction introduces complexities into a mechatronic system. In constructing the model of an existing machine, friction may exist in some location of the system that has not been detected by the user, or the user has not been able to identify it. The "Add Friction" construction function may be used in the course of genetic programming to account for the existence of friction in a modifiable joint. The friction is modeled as a source of effort; however, it is a negative source and should be determined based on the direction of motion. The Stribeck model shown in Figure 6.13 (Tafazoli *et al.,* 1998; De Silva, 2006) is used for friction. If the speed is zero (or, very small in numerical computations) the friction is equal to the force applied to the slider, but in the

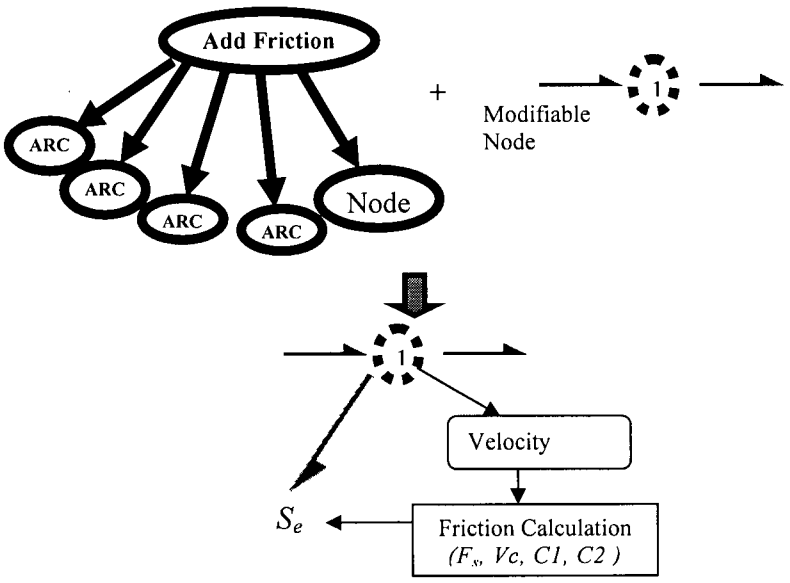


Figure 6.14: Add friction construction function.

opposite direction. However, the friction model needs to be identified, because the corresponding parameter values are unknown. This function has five arguments (Figure 6.14). The first four are arithmetic sites to determine the required numerical values for the Stribeck model (see Figure 6.13). The last one is the old modifiable node site for further modification.

6.2.2.7 Add Backlash Function

Backlash is another source of nonlinearity in a mechatronic system, which is commonly neglected during analytical modeling. If neglected, it can cause undesirable responses, chattering, and even unstable behavior.

A construction functions is developed in the present work in order to examine the possible existence of backlash at a junction of a model. The "Add Backlash" function introduces backlash into a modifiable joint. The backlash between two objects can be modeled as a nonlinear spring (Figure 6.15). The "Add Backlash" function is treated similar to the *Add-C* function which adds a capacitor element (like spring), but the added element is nonlinear in this case. If the relative displacement between the two nodes in backlash is less than the backlash limit, the spring stiffness is zero (no capacitor), and if the relative displacement exceeds the backlash value, a high stiffness spring is considered between the corresponding nodes. Therefore, it has four arguments similar to the *Add-C* function. The first argument is an arithmetic site indicating the amount of backlash and the other arguments are similar to those in the *Add-C* function.

6.2.3 First Generation

First generation of the population is created by applying random combinations of construction functions on the embryo. Tables 6.2 to 6.4 show the available functions for each type of

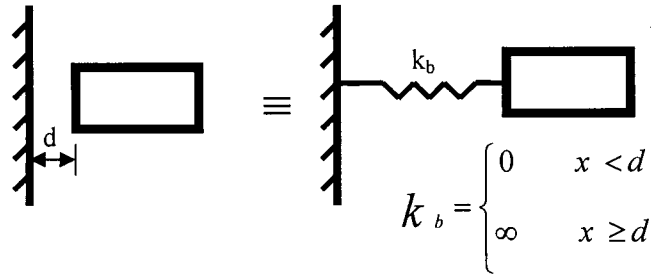


Figure 6.15: Equivalence system of backlash.

modifiable site. The required arguments and the code for each function are also shown in these tables. The left digit in the code corresponds to the number of the arguments of the function. This is useful for vector presentation of tree-like construction functions and for finding the start and the end of each branch.

Table 6.2: Construction functions available for a modifiable node.

Function	Code	Arguments
Add-R	41	4
Add-I	42	4
Add-C	43	4
Add-Backlash	44	4
MDOF	61	6
End-node	2	0

Table 6.3: Construction functions available for a modifiable bond.

Function	Code	Arguments
Insert 0	30	3
Insert 1	31	3
End-bond	-2	0

Table 6.4: Construction functions available for an arithmetic modifiable site.

Function	Code	Arguments
Addition	21	2
Subtraction	22	2
Product	23	2
Division	24	2
Power	25	2
Sign	11	1
Exp	12	1
Log	13	1
A random number		0

For each modifiable site in the embryo, a random number is created by the program. Depending on the value of the random number and predefined ranges for each available function, one of the available functions is selected and is applied on the modifiable site. For example, if there is a modifiable node in the embryo model, one of the *Add-R, Add-C, Add-I, Add-MDOF, Friction, Backlash,* or *End-Node* functions is selected based on the value of the randomly generated number and is applied to the modifiable node. If *End-Node* is selected, then the process for this modifiable site is complete and the program can go to the next modifiable site. If one of the other options is selected, then the applied function creates some inner modifiable sites. This reproduction can continue like branches of a tree. However, the probability for *End-Node* should be high enough to prevent an infinite loop.

This tree-like expansion is the reason that each created GP model is usually called a genetic programming tree. This tree-like expansion feature combined with mutation and crossover genetic operations is the reason for high search ability in genetic programming.

6.3 Automated System Identification

Integration of BG and GP, as developed in the present work, is a powerful approach which can be used for design, optimization, and identification of a mechatronic system. The motivation behind the use of this integration for system identification is the utilization of the power of GP in order to automatically create the model of a system. Most of the time, it is very difficult to find the magnitude of some physical parameters in a mechatronic system. It may even be difficult to exactly specify the topology of the system. The search power of GP, flexibility of BG, and their integration in the developed tool, can be used to find the model of a system automatically. The main steps of the process are as follows:

1-An embryo model for the system is created.

2-Some experiments are performed on the system and the behavior of the system is saved.

3-A fitness evaluation method is selected. Here, the fitness should indicate the degree of closeness of the response of each model to the actual behavior of the system.

4-GP is tuned for this particular task

5-GP is used to evolve the embryo to an accurate model.

Genetic programming is based on Darwin's principle of survival of the fittest. Therefore, fitness evaluation plays a fundamental role in genetic programming. It should adequately

represent the degree of satisfaction of each optional solution. If the fitness evaluation is not satisfactory, then the genetic program may delete good solutions during the genetic programming operation, resulting in inaccurate results.

The approach of fitness evaluation is specific to the problem at hand. For system identification in particular, the fitness of each candidate model may be taken as the degree of closeness of the response of a candidate model to the response of the real system. This comparison can be either in the time domain, the frequency domain, or a combination of the two.

The idea of least squares difference is used in the present thesis, for model comparison. Specifically, to evaluate a parameter denoted by ω, the following expression is used as the fitness function:

$$f = \frac{M}{M + \sqrt{M} \cdot \sqrt{\sum_{k=1}^{M} \left(\frac{\omega_k - \omega'_k}{\bar{\omega}} \right)^2}} \tag{6.4}$$

where, M is the number of comparison points, ω is the actual response of the system, ω' is the response of the model, $\bar{\omega}$ is the average of the absolute values of ω, and the subscript i is an index denoting a particular comparison point. This expression is found to be somewhat independent of the number of comparison points used. If for example, all data have a constant deviation from the actual measurement, then this equation gives a unified f, regardless of the number of comparison points.

6.3.1 Evolutionary System Identification of a Vibratory System

The objective of the example given here is to clarify the power of the developed tool for both sizing and topology optimization. In this example, the developed tool is employed to find the bond graph model of a nonlinear vibratory system. The system has been tested with a step excitation force and the vibration has been measured at the same point where the force was applied.

In this illustrative example, time domain comparison is performed for both displacement and velocity profiles of a vibratory system, using following equation:

$$f = a. \frac{M}{M + \sqrt{M} \cdot \sqrt{\sum_{k=1}^{M} \left(\frac{x_k - x'_k}{\bar{x}} \right)^2}} + b. \frac{M}{M + \sqrt{M} \cdot \sqrt{\sum_{k=1}^{M} \left(\frac{v_k - v'_k}{\bar{v}} \right)^2}} \tag{6.5}$$

where, a and b are two weighting factors, and $\bar{x}$ and $\bar{v}$ are the averages of the absolute values of the displacements and the velocities of the real system, respectively.

The embryo model used here is a simple one-degree-of-freedom (1-dof) mass-spring-damper system, with initially linear elements. The spring is considered as the element with possible nonlinearity. The tool is used to identify the actual model of the system and also a function that can describe the nonlinear element in the system. Initially, the response of the embryo model is quite far from that of the real system. The developed tool was used to evolve this embryo model into a suitable model for describing the real system.

There is a significant difference between using GP for system identification and using it for design purposes. In a typical design problem, GP has to search for a solution that has a response close to the actual response expected from the particular system. There can be several trial designs that have responses very close to each other, while their topologies may be quite different. In a design problem, these solutions will have fitness values close to each other. In contrast, genetic programming has to search for a unique solution in a system identification problem. In the present work it is found that GP may converge to a solution that has a response close to the real system, while its topology is quite different. This situation is more obvious in the identification of a nonlinear system. In fact, GP may converge to a rather complicated solution to create a response close to the response of the real nonlinear system. One reason for this situation is that all the construction functions have an additive effect.

In addressing this problem, it is found helpful to incorporate knowledge in the GP optimization process. Any knowledge from the user can assist GP to converge easier and faster to a better solution. In this particular example, it is assumed that the user knows that the mechanical degree of freedom of the system is unlikely to be greater than three. This knowledge is incorporated into the GP process by incorporating a penalty factor for high degrees of freedom in a trial solution.

A result obtained by using the developed tool in the present example is shown in Figure 6.16, which is quite satisfactory. Figure 6.17 shows the fitness history and its improvement. The interesting point to note is that the tool has effectively determined that a 2 degree-of-freedom system is necessary to model the system, and has optimized both size and topology of the embryo model to produce good results. Figure 6.18 shows the embryo model and the evolved model. This example illustrates the power of the developed tool for both sizing and topology optimization for a nonlinear problem.

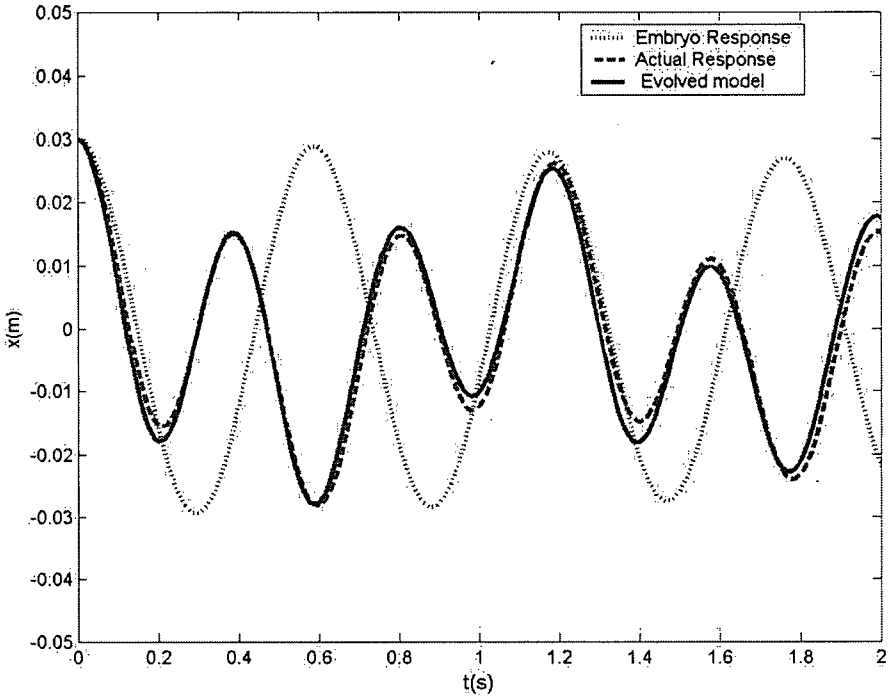


Figure 6.16: The performance of the developed tool in evolutionary modeling of a system.

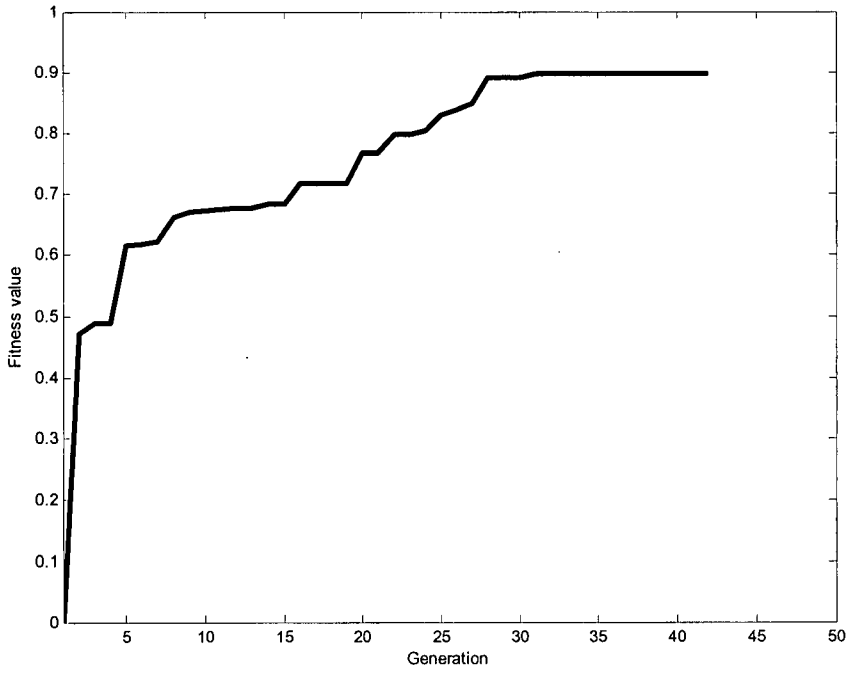
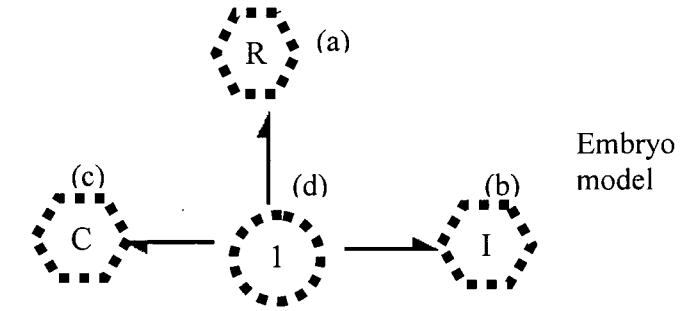
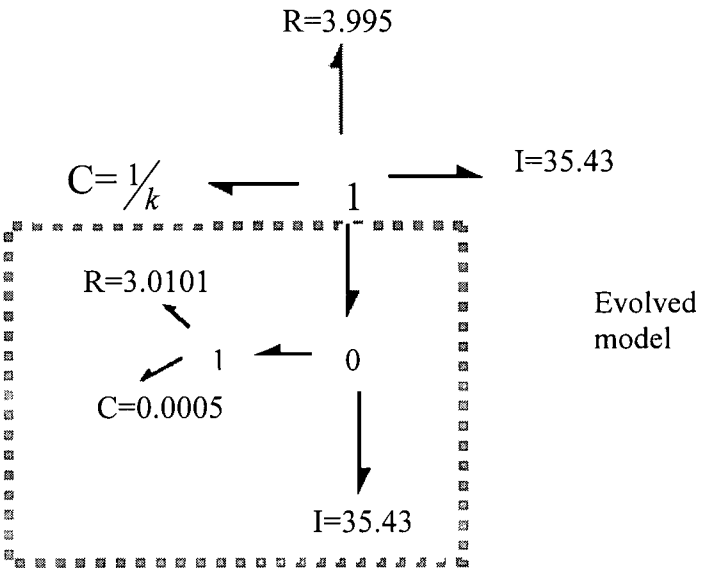


Figure 6.17: Fitness improvement using genetic programming.





$$k = 6880 + \left[sign(x) + x^{2} \right] \left[x . \log\left(\frac{1.9186}{x + 2.4596} \right) \right]$$

X=displacement of the first node

Figure 6.18: Embryo and evolved model in evolutionary system identification of a vibratory system.

6.3.2 System Identification of an Electro-Hydraulic Manipulator

A novel machine has been designed and built in the Industrial Automation Laboratory of the University of British Columbia (UBC) to automatically cut the heads of fish with maximum meat recovery (Tafazoli *et al.*, 1998; De Silva, 2005). The location of the gill of a fish is found by a vision system and is sent to a positioning table to optimally locate the cutter. The positioning table is actuated by two electro-hydraulic manipulators with two servovalves. A schematic diagram of the electro-hydraulic system is shown in Figure 6.19. This electro-

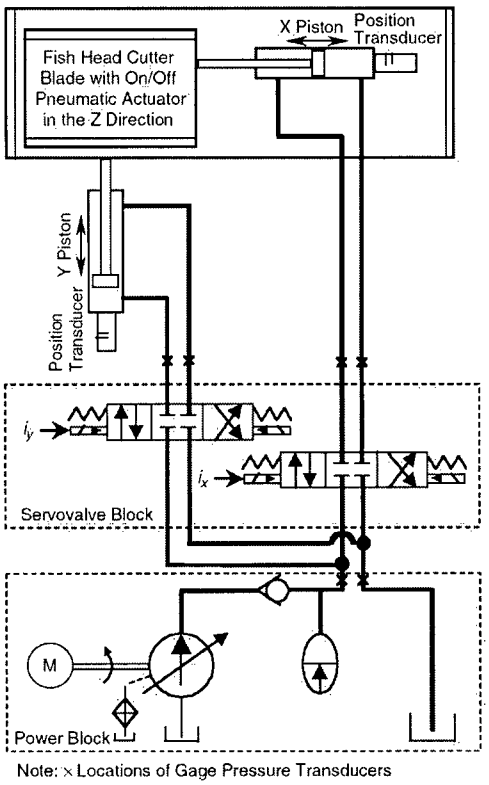


Figure 6.19: Schematic diagram of the electro-hydraulic manipulator.

hydraulic system is also used as the test bed for several research projects, which are aimed at developing advanced control technologies to cope with such factors as nonlinear friction, system parameter variation, and communication time delays. An accurate and comprehensive model is quite useful in controlling this system, which is highly nonlinear. Important parts of the system are:

1- Variable displacement pump
2- Servovalve
3- Hydraulic cylinder
4- Mechanical subsystem including the cutter carriage table and the guideways.

Let us now consider the development of a bond graph model for this system. The procedure for bond graph modeling of mechanical, electrical and hydraulic components is found in (Karnopp *et al.,* 2000). Figure 6.20a shows a schematic diagram of the hydraulic cylinder. Following the standard procedure, the bond graph model of this component is developed, which is shown in Figure 6.21a. The leakage between the two sides of the piston, through the cylinder wall, is accounted for by placing a resistance in the leakage path (Cellier *et al.*), between these

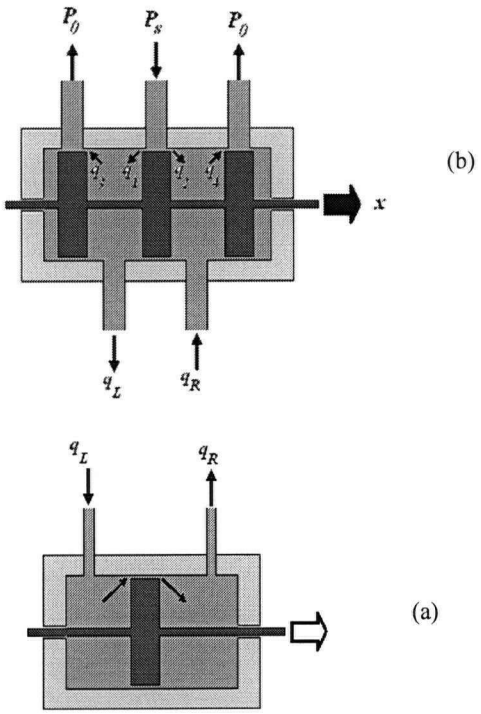


Figure 6.20: Schematic diagram of different parts of the electro-hydraulic manipulator (Cellier *et al.*).

two sides. The resistance should be fairly large, to restrict the leakage. The capacitors in the two sides of the cylinder come from the compressibility of the fluid and the flexibility of the cylinder, including possible spring loading on the piston (De Silva, 2005). In bond graph terminology, there exists a constitutive law between the effort variable, *e,* and the flow variable, *f*; for example, voltage and current of an electrical capacitor or force and velocity of a mechanical spring (flexibility corresponds to capacitance):

$$C\frac{de}{dt} = f \Rightarrow e = \frac{q}{C} \tag{6.6}$$

In BG terminology, q is the time integral of a flow variable and is called the displacement variable (for example, electrical charge, mechanical displacement, and fluid volume in electrical, mechanical, and hydraulic systems, respectively). In a hydraulic system, pressure (P) represents the effort variable, the volume flow rate is the flow variable, and fluid volume is the displacement variable. Considering the parameters shown in Figure 6.22, the capacitor coefficients can be estimated from the compressibility equation (Viersma 1980) as:

$$C = \frac{A.S + V_L}{4A.E} \tag{6.7}$$

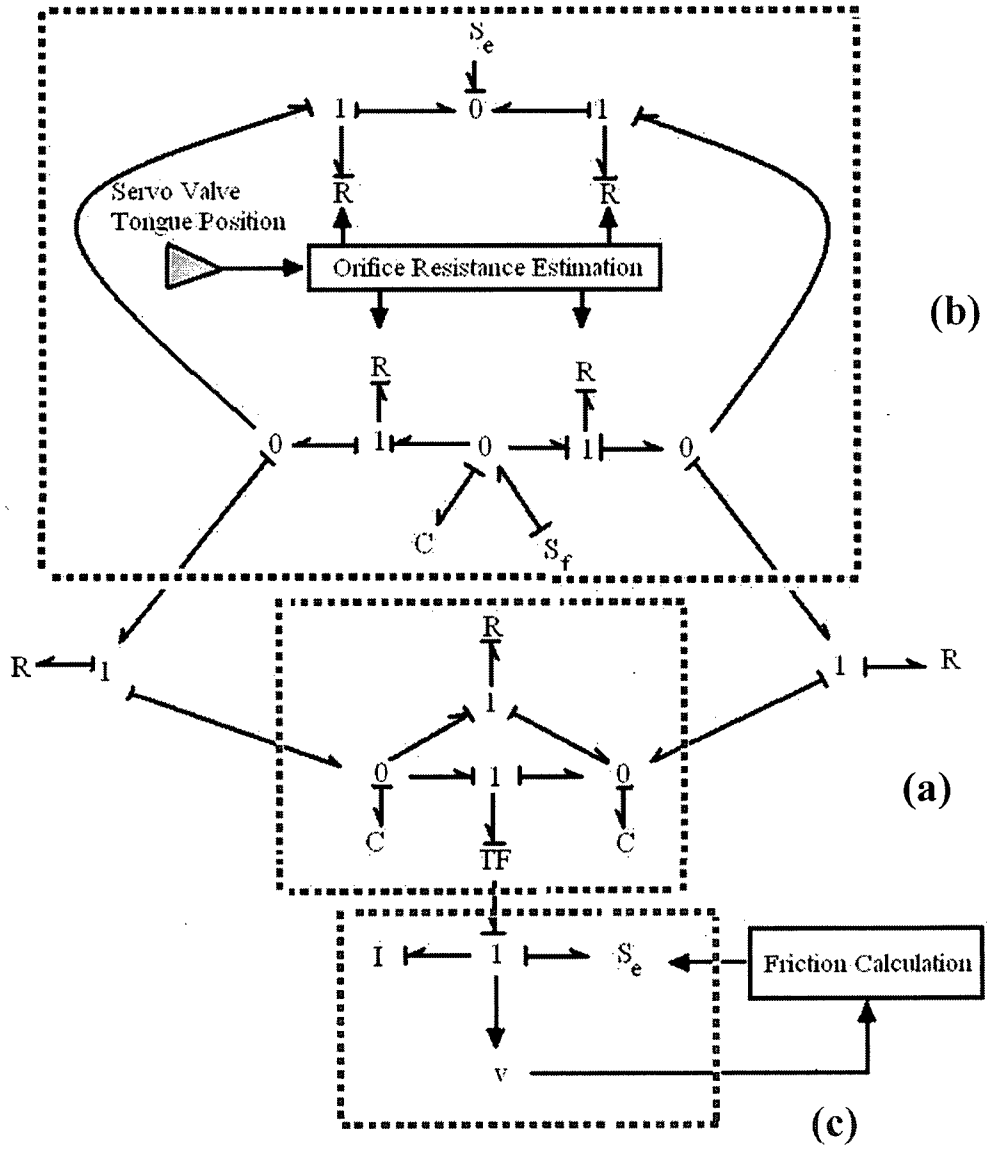


Figure 6.21: The embryo bond graph model for electro-hydraulic manipulator.

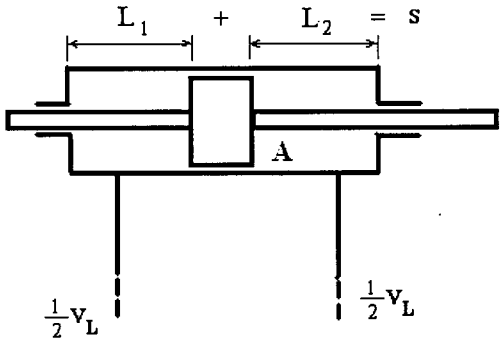


Figure 6.22: Schematic diagram of the hydraulic cylinder.

Figure 6.20b shows the schematic diagram of the servovalve. Depending on the position of the tongue, four orifices are formed in the servovalve (Cellier *et al.*). These four orifices can be modeled as four nonlinear resistances. The bond graph model of the servovalve is shown in Figure 6.21b. Depending on the position of the tongue of the servovalve, the magnitude of the four generated orifices should be estimated and be substituted into the model. In other words, these four orifices should be represented by functions of the position of the tongue of the servovalve.

The mechanical subsystem is simply a sliding mass subjected to friction. The bond graph model of the mechanical subsystem is shown in Figure 6.21c. The friction has been modeled by the Stribeck model (Tafazoli *et al.,* 1998) as a source of effort; however, it is a negative source and should be determined based on the direction of motion.

The tongue of the servovalve has a controller, which is a low-power electrical circuit. It does not have significant interaction with the other parts of the system, and can be modeled separately. Figure 6.23 shows the schematic diagram and the bond graph model of this subsystem (Cellier *et al.*).

By assembling different subsystem models, the embryo model of the system is obtained (Figure 6.21). Nonlinearity of this system mainly comes from the friction under the slider and from the hydraulic resistance of the four orifices in the servovalve. In this example, the evolutionary tool was used to find the magnitudes of all the elements of the model shown in Figure 6.21, identify friction, and find functions representing the variation of the nonlinear resistances of the orifices in the servovalve. An experiment was carried out on the machine by applying a sine wave excitation. The fitness evaluation is performed by comparing a combination of pressure and displacement profiles, both in the time domain and the frequency domain. Figure 6.24 and 6.25 compares the displacement and the pressure response of the evolved model with the response of the actual physical system. It is noticed that the agreement is quite satisfactory, thus validating the model. The main reason for the slight deviation between them is the random nature of the friction. In this system, it is found that the friction represents a higher order than the inertia forces (nearly 20 times more). It means that a small deviation in the prediction of the friction is expected to cause a significant deviation in the displacement and velocity profiles. Since the real friction has a random characteristic, a deviation between the real friction and the predicted friction is inevitable. Therefore, the agreement between the responses of the created model and the real system is quite satisfactory, despite the slight deviation that is observed between them.

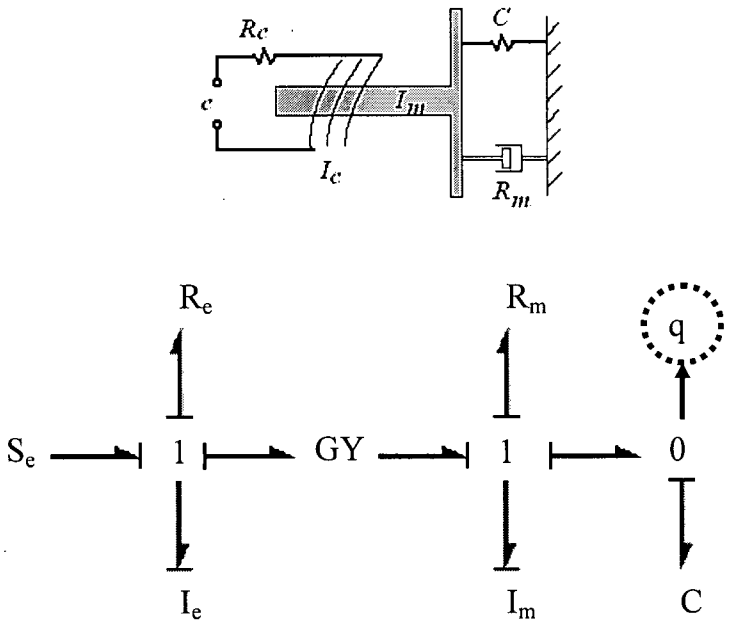


Figure 6.23: Schematic diagram and the bond graph model of the servo valve tongue controller (Cellier, *et al.*).

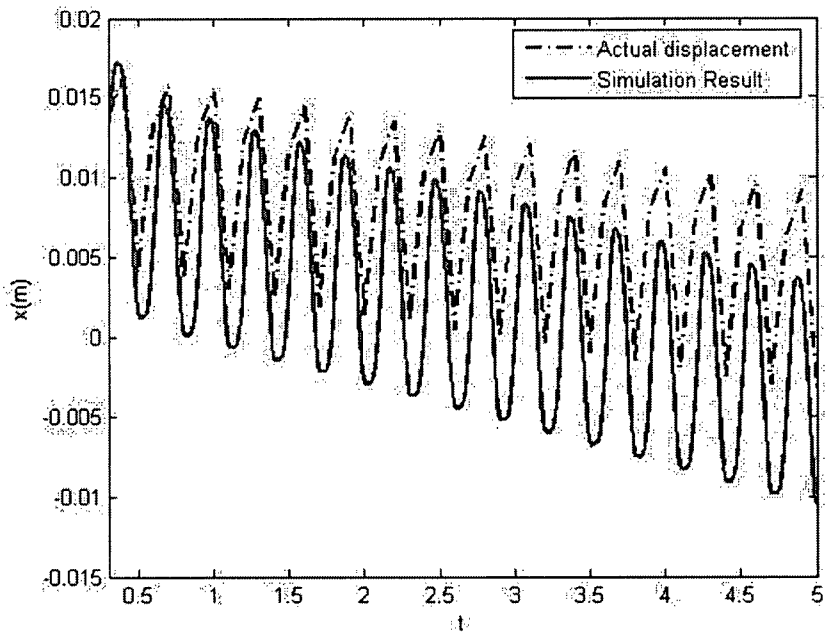


Figure 6.24: Displacement comparison for system identification of electro-hydraulic manipulator.

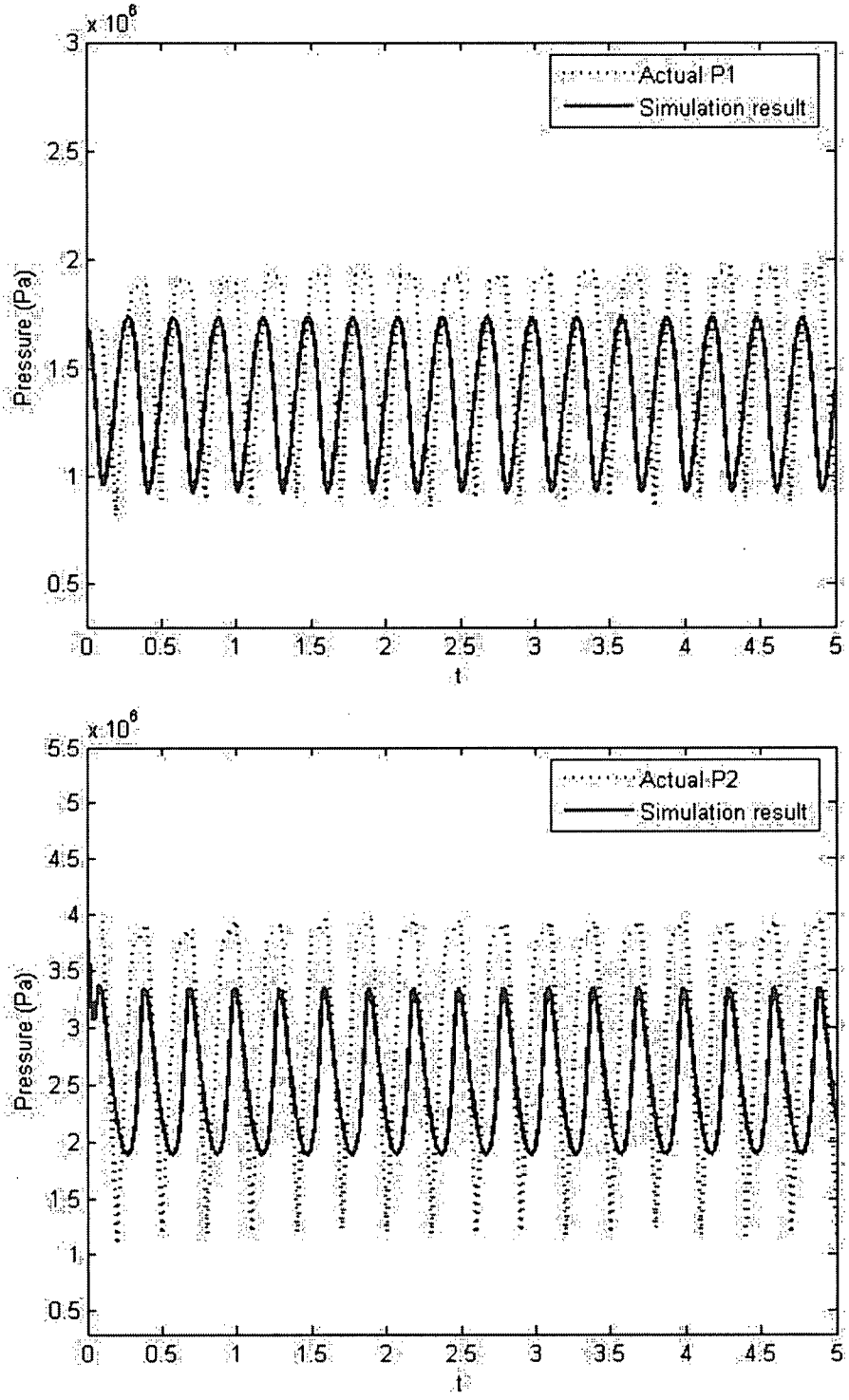


Figure 6.25: Pressure comparison for system identification of electro-hydraulic manipulator (P1 and P2 are the pressures at two output points of the servo valve).

6.4 Summary

Modeling of a mixed mechatronic system is a challenging task due to the presence of complex subsystems in different domains and the need to integrate different engineering fields, in representing the overall mixed system. A bond-graph-based evolutionary system tool was developed to model and identify mixed systems in a rather optimal manner. Integration of bond graph and block diagram modeling provided a domain-independent simulation environment suitable for a mixed system. Genetic programming was linked to this environment to create a multi-purpose evolutionary tool. This tool can be used for model identification in a variety of problems, provided that a suitable evaluation scheme can be established for that problem. In the first stage, the tool was applied successfully for system identification. An electro-hydraulic manipulator of a fish processing machine, which is a highly nonlinear mechatronic system, was identified by this tool to show its feasibility and performance. The obtained results were quite encouraging, leading to a rationale for extending the tool for the development of a more general mechatronic design tool. A rather challenging issue would be the establishment of a formal methodology for evaluation of mixed systems. Intuitively, an evaluation methodology for mixed systems should cover a wide range of criteria and cannot be summarized in a simple equation. Conventional optimization tools are not capable of handling such problems in an effective manner. Genetic programming should provide a desirable alternative for evaluation and optimization of a model or a design, resulting in a practical tool to handle multi-criteria mechatronic modeling and design, as demonstrated in the present chapter.

Chapter 7

Reliability Assessment Using Petri-Net Approach

A typical mechatronic system consists of several subsystems in different engineering domains such as mechanical, electrical, hydraulic, control, digital hardware, and software. These subsystems have operational dependency with each other. The actuator components rely on the measurements provided by the sensory components and the computed command signals provided by the controller components. Failure may occur in any of these components, or a particular failure mode may be caused by undesirable interactions between the components (Guérin *et al.*, 2002).

An optimal approach for mechatronic design is expected to improve the reliability of the designed system/product. Accordingly, reliability has to be considered an essential criterion in the design process and the associated decision making. In this manner, the designer needs to assess the reliability of the design choices in different stages of the design process.

In this chapter, a formal and systematic methodology for reliability assessment in repairable mixed-system (multi-domain) machinery is presented. The performance of the developed methodology is validated by applying it to an industrial fish cutting machine-- the Iron Butcher, which falls into the class of multi-domain systems. Methods to improve the reliability of a mixed system are represented and the performance of the proposed methodology is validated by demonstrating its ability to reflect axiomatic expectations (Behbahani and de Silva, 2006e).

Section 7.2 of this chapter presents the developed approach. It also discusses the reliability of repairable machines and offers an analytical expression to represent it. In section 7.2.1 a fuzzy framework for the assessment of failure severity is presented. Section 7.2.2 extends the concept of Risk Priority Number (RPN) to the reliability assessment technique developed in the chapter. Sections 7.2.3 and 7.2.4 describe how to model failure and failure recovery events. Section 7.3 presents a case study and validates the developed approach by applying it to a practical mechatronic system.

7.1 The Role of Reliability Assessment in Mechatronic Design

In using the mechatronic design methodology developed in this thesis, there are several points where the designer needs to make crucial decisions about the architecture of the machine which is being designed. One of them is in the detailed design stage, after a niching genetic algorithm

gives the elite representatives of different possible configurations. Since all elite design choices have satisfactory performance, the designer needs to perform a very crucial comparison between them to find the best.

Another point concerns the final improvement stage. There exist several techniques and architectural extensions to improve the overall performance of a machine, such as the use of high-level supervisory control, fault detection and diagnosis methodologies, scheduling policies, repair and maintenance policies, and component redundancy. These features can have a significant impact on the performance of the machine, but they need more resources and the cost of the machine will increase. As a result, for each improvement technique, the designer needs to precisely evaluate the possible performance improvement from that scheme and trade off with the cost increase to see whether it is justifiable to implement it.

Reliability of a mechatronic system is one of the essential criteria in making design decisions. Mechatronic design is expected to result in higher reliability which implies lower probability of failure, a degree of intelligence for taking proper actions in case of failure, and lower harm due to occurrence of failure. A proper assessment of reliability in different stages of design is necessary in order to take reliability into account as a design criterion. Classical methods such as fault tree analysis (FTA), which are static, may not be adequate to deal with complex and hybrid dynamic systems that fall in the category of mechatronic systems (Demmou *et al.,* 2004). On the other hand, common reliability specifications such as time to first failure (TTFF) and time between failures (TBF), for repairable systems (Blischke *et al.,* 2003), are not intuitive because they do not consider the severity of the failure modes.

This chapter presents a formal and systematic methodology for reliability assessment in repairable mixed-system (multi-domain) machinery based on an intuitive severity assessment of failure modes using Choquet fuzzy integral. A simulation technique based on Petri nets is used to mimic the dynamic behavior of the system and to take into account dynamic interactions between the system components. The method has two advantages compared with common techniques of reliability assessment, as indicated next.

First, the method is able to clearly reflect dynamic interactions through Extended Stochastic Petri-Net (ESPN) simulation. This approach mimics the dynamic operation of a large set of copies of the machine and records the history of failure modes, cost of repair, and repair time for their operating life. The Petri-net model of the dynamic structure of a mechatronic system includes all possible *states* and *events* in the actual operation of the system, including normal operating conditions, events leading to critical and feared states, and repair events (Dutuit *et al.,*

1997; Guérin *et al.*, 2002). Similar to the actual operation of the system, different events are fired sequentially depending on the system architecture and by considering the stochastic execution time for different events. Some events may have conflicts with each other which may be resolved by specifying the probability values for the conflicting events. In Petri-net terminology, a model that includes the stochastic execution time of events is called an Extended Stochastic Petri-Net (ESPN) (Ireson *et al.*, 1996).

Second, the method is able to reflect intuitive reliability expectations and reliability feelings, by taking into account the severity of failure modes. A framework is developed in this chapter to assess the severity of each failure mode. Choquet fuzzy integral is used to aggregate the severity attributes for displaying the feelings of the reliability analyzer.

7.2 Reliability Assessment of a Mechatronic System

Reliability is time dependent due to the dynamics of the system architecture, aging and debugging effects, and other time-varying factors. An ideal way to assess the reliability of a machine would be to let several copies of the machine operate for the design life of the machine (say 50 years) and record the history of failures, cost of failures, and harm caused by the failures, and then to analyze this data. However it is not a practical method, particularly in the design stage, and hence it is not useful in mechatronic design. Fortunately, the processing power of modern computers and the modeling power of Petri nets have made it possible to mimic this idealized assessment procedure in a matter of minutes.

With regard to reliability specification, machines can be divided into two categories: non-repairable machines and repairable machines. The concept of reliability is more controversial for a repairable system than for a non-repairable system. For a non-repairable machine, as soon as a failure occurs, the entire machine is considered failed. This may be the case because, for example, the failed device is not worth repairing as for most electrical printed circuit units, or the failed system is not easily and economically accessible for repair as for a satellite. Therefore, a non-repairable machine can be judged whether it has failed or not without any controversy. Reliability can be evaluated by using the formula:

$$R(t) = \frac{n_h(t)}{N} = 1 - \frac{n_f(t)}{N} \tag{7.1}$$

where $n_h(t)$ is the number of machines still in operation, $n_f(t)$ is the number of failed machines and $N = n_h(t) + n_f(t)$ is the initial number of machines. This formula provides a probability distribution for time to failure (De Silva, 2006).

For a repairable machine, such a straightforward reliability definition cannot be offered. A complex repairable system does not have a time-to-failure distribution because it may fail quite frequently and be repaired and used again and again. In contrast, some other specifications are used in literature such as probability distributions for time-to-first-failure (TTFF) and time-between-failures (TBF) (Blischke *et al.*, 2003). Although these specifications can give a sense of reliability satisfaction for a repairable machine, it can be argued that they are not quite appropriate.

A good example of a repairable machine is an automobile. During the life time of an automobile, different failures may occur and be repaired, and the automobile may be used again and again after each repair. These failures can cause different levels of dissatisfaction for the owner. For example, a simple burning of a light bulb does not cause a serious dissatisfaction when compared to the failure of the transmission system, while both are component failures. Some failures may seem quite normal to the user and will not cause significant reliability dissatisfaction. The specifications TTFF and TBF do not discriminate between different levels of failure, and as a result do not provide a good sense of reliability for a complex repairable machine.

The method of reliability assessment developed in this chapter considers the severity of different failure modes in the specification of the reliability of a repairable machine. The dissatisfaction due to each failure mode is compared with the dissatisfaction due to failure of the entire machine. A failure fraction is assigned to each machine at any time t.

With reference to equation (7.1), $n_f(t)/N$ may be interpreted as a penalty function which is subtracted from the ideal machine reliability each time that one sample machine fails. If one of the machines fails completely, then a penalty of $1/N$ is subtracted from the reliability of the machine. In a repairable system, the occurrence of a failure mode cannot be interpreted as a complete failure of the machine. Instead, a particular failure mode may be considered a partial failure of the overall system. The machine reliability should then be penalized based on the severity of the incurred failure mode. This leads to the consideration of a severity factor for each failure mode. Reliability can then be represented by:

$$R(t) = 1 - \frac{\sum_{i=1}^{N} \tilde{f}_i(t)}{N} \tag{7.2}$$

where $\tilde{f}_i(t)$ represents the partial failure value of the i^{th} machine in time t. It is clear that $\tilde{f}_i(t)$ is always less than one. It can be computed using:

$$\tilde{f}_i(t) = \min\left\{1, \sum_{j=1}^{NF} S_j \times n_{ij}(t)\right\} \tag{7.3}$$

Here NF is the number of possible failure modes, S_j is the failure severity of the j^{th} failure mode, and $n_{ij}(t)$is the number of times the j^{th} failure mode has happened in the i^{th} sample machine up to time t.

7.2.1 Severity Assessment of a Failure Mode

As indicated before, different failure modes may not be equal in terms of their severity. Different factors will contribute to the severity of a failure mode, in particular (Figure 7.1):

1)Safety

2)Operational dependence/interaction

3)Detection and diagnosis procedures

4)Repair cost

5)Downtime

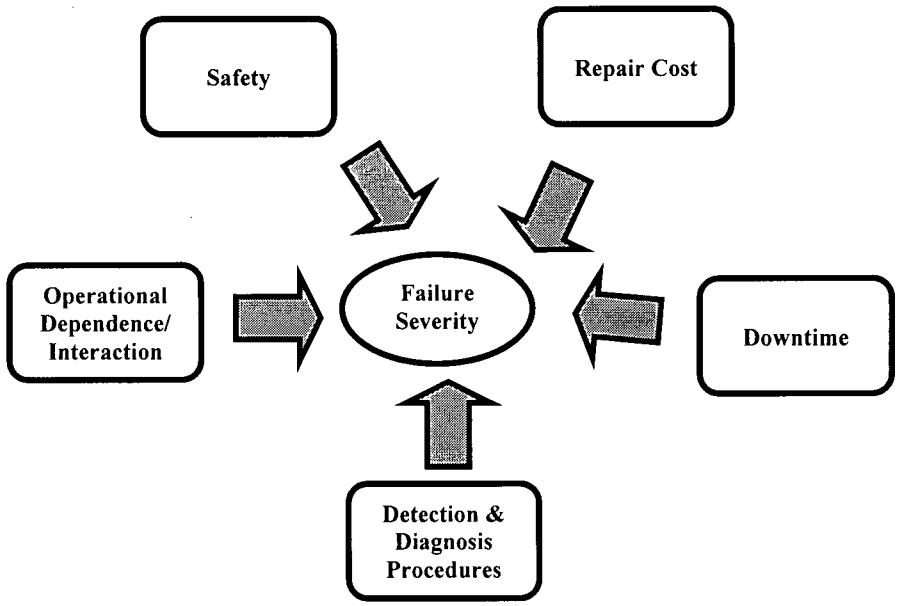


Figure 7.1: Criteria of severity assessment.

For each failure mode, a partial severity score between zero and one is assigned to each of these listed items. Tables 1 to 4 give guidelines for specifying the partial scores of severity of a failure mode.

Table 7.1: Classifications for the safety concerns of a failure mode.

Condition	S_{j1}
Very hazardous, without alarm, without periodical inspection	1
Very hazardous with periodical inspection	0.9
Very hazardous, with automatic alarm system	0.8
Hazardous, without alarm, without periodical inspection	0.7
Hazardous with periodical inspection or alarm system	0.6
Medium hazard, without alarm, without periodical inspection	0.5
Medium hazard with periodical inspection or alarm system	0.4
Low hazard, without alarm, without periodical inspection	0.3
Low hazard with periodical inspection or alarm system	0.2
No significant hazard	0.1
No safety concern at all	0

Table 7.2: Classifications for operational dependence/interaction of a failure mode.

Condition	S_{j2}
Extremely high, a large number of machines are prevented from normal operation	1
Very high, loss of quality for a large number of machines	0.9
High, the entire machine stops working, a bottleneck for a set of machines	0.8
High, the entire machine stops working	0.7
Medium-to-high, the machine loses its primary function, but some other functions can still be performed	0.6
Medium, the machine loses the primary task, but can be used for other tasks without any shortage	0.5
Medium, the machine loses a function which is not primary and seldom is used	0.4
Low, a quality degradation in a secondary function	0.3
Low, a small part of the machine loses its functionality	0.2
Very low, no effect because of active redundancy	0.1
Not at all	0

Table 7.3: Classifications for detection and diagnosis difficulty of a failure mode.

Condition	S_{j3}
Absolute uncertainty	1
Very remote chance	0.9
Remote	0.8
Very low	0.7
Low	0.6
Moderate	0.5
Moderately high	0.4
High	0.3
Very high	0.2
Almost certain	0.1
Completely certain	0

Table 7.4: Classifications of repair cost of a failure mode.

Condition	S_{j4}
Not worth repairing	1
Hardly worth repairing	0.9
Very high cost	0.8
High	0.7
Moderately high	0.6
Moderate	0.5
Low	0.4
Moderately low	0.3
Very low	0.2
Not considerable	0.1
No cost	0

Table 7.5: Classifications of repair time of a failure mode.

Condition	S_{j5}
Very time consuming, not worth repairing	1
Hardly worth repairing due to very high repair time	0.9
Very highly time consuming	0.8
High repair time	0.7
Moderately high repair time	0.6
Moderate repair time	0.5
Low repair time	0.4
Moderately low repair time	0.3
Very low repair time	0.2
Not considerable repair time	0.1
Negligible repair time	0

The severity of a failure mode is then computed by aggregating these partial scores as

$$S_j = H(S_{j1}, S_{j2}, \ldots\ldots, S_{jn}) \tag{7.4}$$

where H is an aggregation operator and S_{ji} is the partial score of the j^{th} failure mode according to the i^{th} criterion.

In section 7.2, Choquet integral is introduced as an intuitive aggregation technique for considering interactions between criteria. The same technique is used here for aggregation of above criteria and for assessment of the severity of a failure mode.

7.2.2 Risk Priority Number (RPN)

Risk priority number is a useful specification for the failure mode and effects analysis (FMEA) addressed in the fault tree analysis (FTA) methodology (Ireson *et al.*, 1996L; Cassanelli *et al.*, 2003). For a particular failure mode, *RPN* is computed by multiplying together its severity, occurrence probability, and its detection ratings. It indicates the design risk, the concern level of a failure mode, and its priority for additional investment leading to quality improvement.

The occurrence probability of a failure mode can be found from Petri-net simulation, according to

$$O_j = \frac{\tilde{n}_j}{\tilde{N}_j} \tag{7.5}$$

where $\tilde{n}_j$ is the number of times the failure mode has occurred and $\tilde{N}_j$ is the total number of times that occurrence of the failure mode has been examined.

7.2.3 Failure Rate Estimation

Failure rate is the basic information that is required for reliability analysis. It is time dependent and also depends on the reliability of the components involved in the failure mode, their age, quality, life expectancy, and load distribution. Life time models are available in reliability handbooks for different types of components, which are quite useful. The most commonly used model for component life time is the Weibull model (Ireson *et al.,* 1996).

$$F(t) = 1 - e^{-\left(\frac{t-\gamma}{\alpha}\right)^{\beta}} \tag{7.6}$$

where, *F(t)* is the cumulative distribution of the population fraction that has failed by age *t*; β is a nondimensional positive number, usually in the range of 0.5 to 5 and is called the *shape parameter*; and α is a positive number with units of time (hours, days, etc.) and is called the *characteristic life*. In this work, the parameter γ is used to represent the repair policy; for example, it represents the last repair time if the component is exchanged.

The failure probability of Weibull distribution function is given by:

$$\lambda = \frac{\dot{F}(t)}{1-F(t)} = \left(\frac{\beta}{\alpha^{\beta}}\right).(t-\gamma)^{\beta-1} \tag{7.7}$$

Failure probability is used for conflict resolution between *healthy* and *failed* transitions in a Petri-net operation. It is plotted in Figure 7.2 for $\alpha = 20$ years and for different magnitudes of β. If $\beta = 1$, the failure probability is constant, indicating that it does not have a significant aging effect. $\beta > 1$ provides an increasing failure probability, which is suitable for modeling the aging effect. $\beta < 1$ represents a decreasing failure probability, which is suitable for modeling debugging ffects like software failure modes.

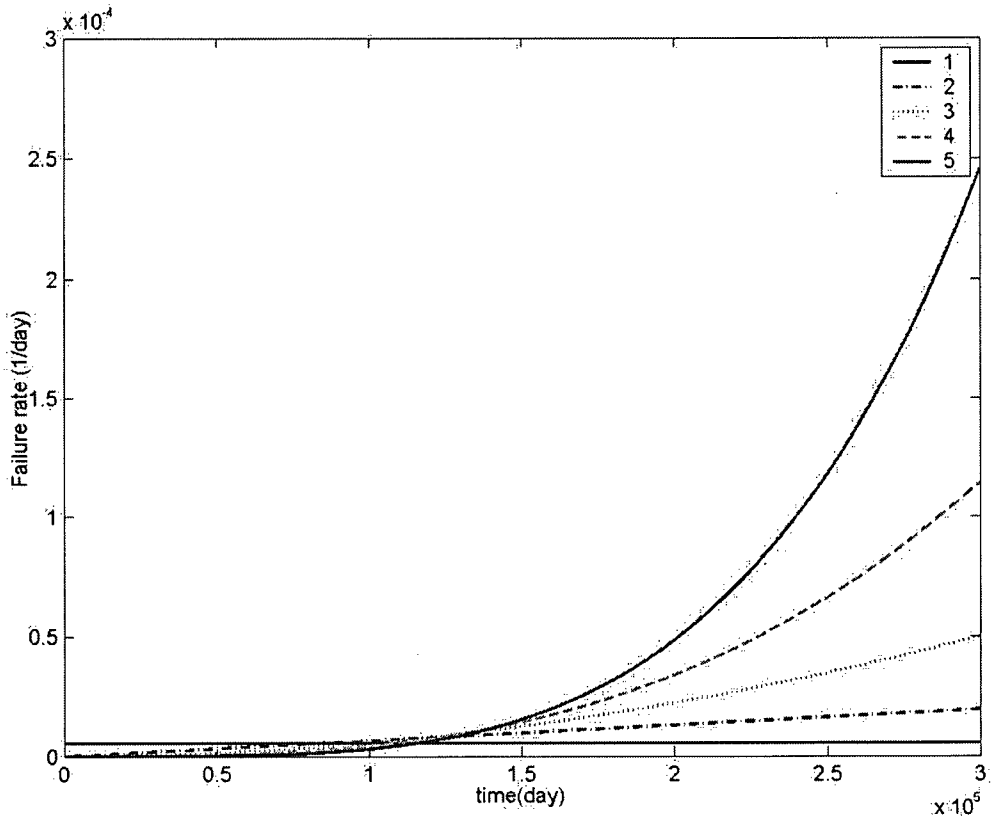


Figure 7.2: Failure rate distribution according to Weibull function for $\alpha = 20$.

7.2.4 Failure and Recovery Modeling

The first step in the reliability assessment of a system (e.g., machine) is to model the dynamic architecture of the system by Petri nets. All possible conditions including normal operation and faulty conditions should be taken into account here. However, it is the designer who should decide on the proper level of detail that would go into the model. Some components may be modeled and analyzed separately and only the results be incorporated into the main model. For some components reliability data may be available in handbooks, literature, or from the manufacturer, which can be incorporated into the main model, thereby simplifying it.

The lowest level of failure is the failure of a component. From the Petri-net point of view, each component can have two conditions: *healthy* or *failed*. However a more detailed model can also be considered by expanding each of these conditions. For example, a failed condition can be elaborated into failed, detected, and diagnosed conditions.

A simple Petri-net model of failure and recovery transitions is shown in Figure 7.3. More information about Petri-net modeling can be found in (Peterson, 1981; Zhou *et al.*, 1999). A component is able to perform its task only if it is in a healthy state, when there is a token in the

"*Component-OK*" location. The "*Component-OK*" is a precondition for the normal transition and it has to be the post-condition as well. This is so because according to execution rules, the tokens in the inputs are removed after firing the transition. Therefore "*Component-OK*" must be the output of the transition as well as its input, to indicate that the component remains healthy after carrying out its task.

If "*Component-OK*" has a token, there will be a conflict between the two transitions: "*kept healthy*" and "*failed.*" It indicates that the model frequently *examines* the health of the component. For a particular model, the designer should determine the frequency of examination of the component. The associated unit of time should be the same as that by which the failure rate is represented. For example, if the failure probability is represented in the units of "per day," the examination should also be performed once a day. Therefore, both conflicting transitions have an input from a '*clock*' which enables them to be executed at the examination frequency.

To resolve a conflict and decide if a component has failed, a random number is created each time that the model examines the component (each time that clock fires the examination process). If the random number is smaller than the failure probability density, the component is considered as failed; otherwise the component is considered to be healthy. In this method, conflicting events are fired stochastically and in proportion to their probability value, similar to what happens in real-life operation of a machine. By applying this strategy on all failure modes and all other conflicting events, one is able to mimic the actual dynamic operation of the system.

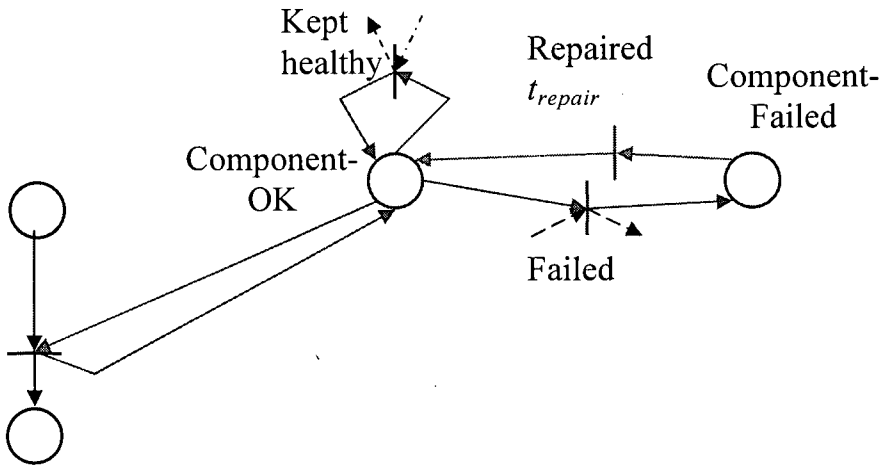


Figure 7.3: Petri-net model for failure and recovery of a component.

As soon as a failure occurs, the *'Component-Failed'* location will get a token and then the *'repaired'* transition becomes executable. Since the repair process takes time, a firing time is associated with this transition. The firing time can be specified by a stochastic distribution so as to mimic the actual condition as accurately as possible. Required time for failure detection, diagnosis and failure recovery are included in this firing time. Different repair policies can be modeled. A more detailed model can include detection and diagnosis as separate states, but all these stages can be considered in the repair process from the reliability analysis viewpoint.

7.3 Reliability Assessment of the Iron Butcher

A machine that is commonly used in the fish processing industry for cutting the fish heads is known as the Iron Butcher (Tafazoli *et al.,* 1998; de Silva, 2005). A prototype of this machine is available in the Industrial Automation Laboratory of the University of British Columbia. In this section a Petri-net model of this machine is presented and a reliability analysis is carried out.

Figure 7.4 shows a schematic diagram of the Iron Butcher. The machine has four subsystems:

A. Fish motion
B. Fish stabilization
C. Cutter vertical motion
D. Cutter horizontal motion

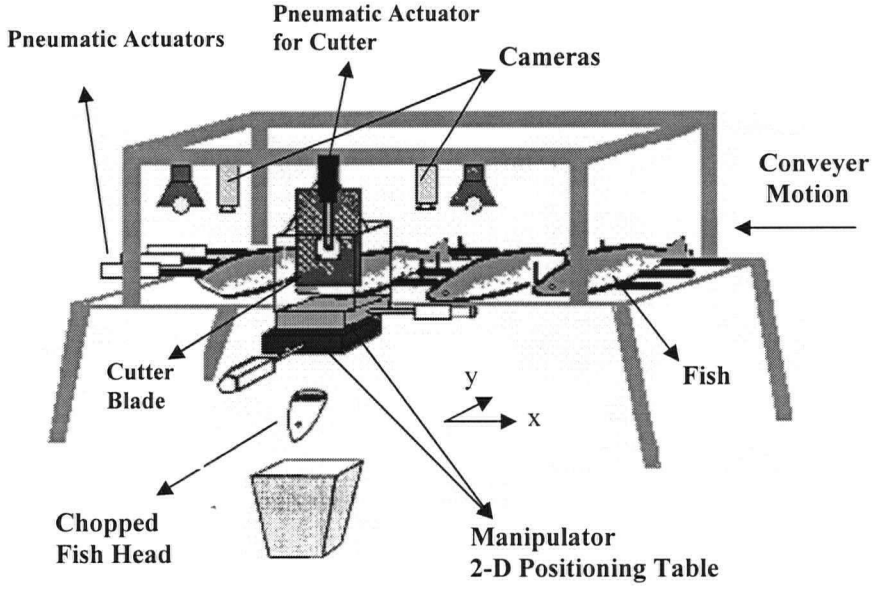


Figure 7.4: Schematic diagram of the Iron Butcher.

System A is designed to trigger systems B and C and also generate the fish motion. The input device is the push button of the machine and the actuator is an induction motor.

Control system of B is designed to stabilize the fish while capturing its image and during cutting. The input device is a binary limit switch triggered by the control system A. The end effector is a stabilizing mechanism which applies a force on the fish.

Control system C generates the motion of the cutter in order to cut the head of a fish. The input device is a binary limit switch triggered by the control system A. The end effector is the zigzag contoured steel cutter blade which is moved by a pneumatic actuator.

Control system D is a highly important and complicated part of the system. It is designed to move the cutter to the desired x-y location (optimum location for cutting) which is determined by a computer vision system. The on-off signal from the proximity sensor is transmitted to the computer vision system to trigger a CCD camera to capture a 2-D image of the scene and send this image data to the host computer for processing. The computer vision algorithm processes the image data, obtains the optimum location for cutting, and sends the corresponding x-y coordinates as a set point to the control system D. Then the task of system D is initiated, which is to engage the cutter at the set point. The actuator is a hydraulic cylinder which generates the manipulator force. A servovalve, which is a control actuator, is also present. It changes the pressure and the flow rate to the hydraulic cylinder. The input signal to the servovalve is the current of the valve actuator, which controls the applied force to the system through changing the pressure and the flow rate of the hydraulic cylinder. The end effector is the manipulator which is attached to the piston of the hydraulic cylinder (Figure 7.5).

Figures 7.5 and 7.6 show the Petri-net model of the entire system and details of the electro-hydraulic manipulator, respectively. Different conditions (places) are explained in Table 7.6.

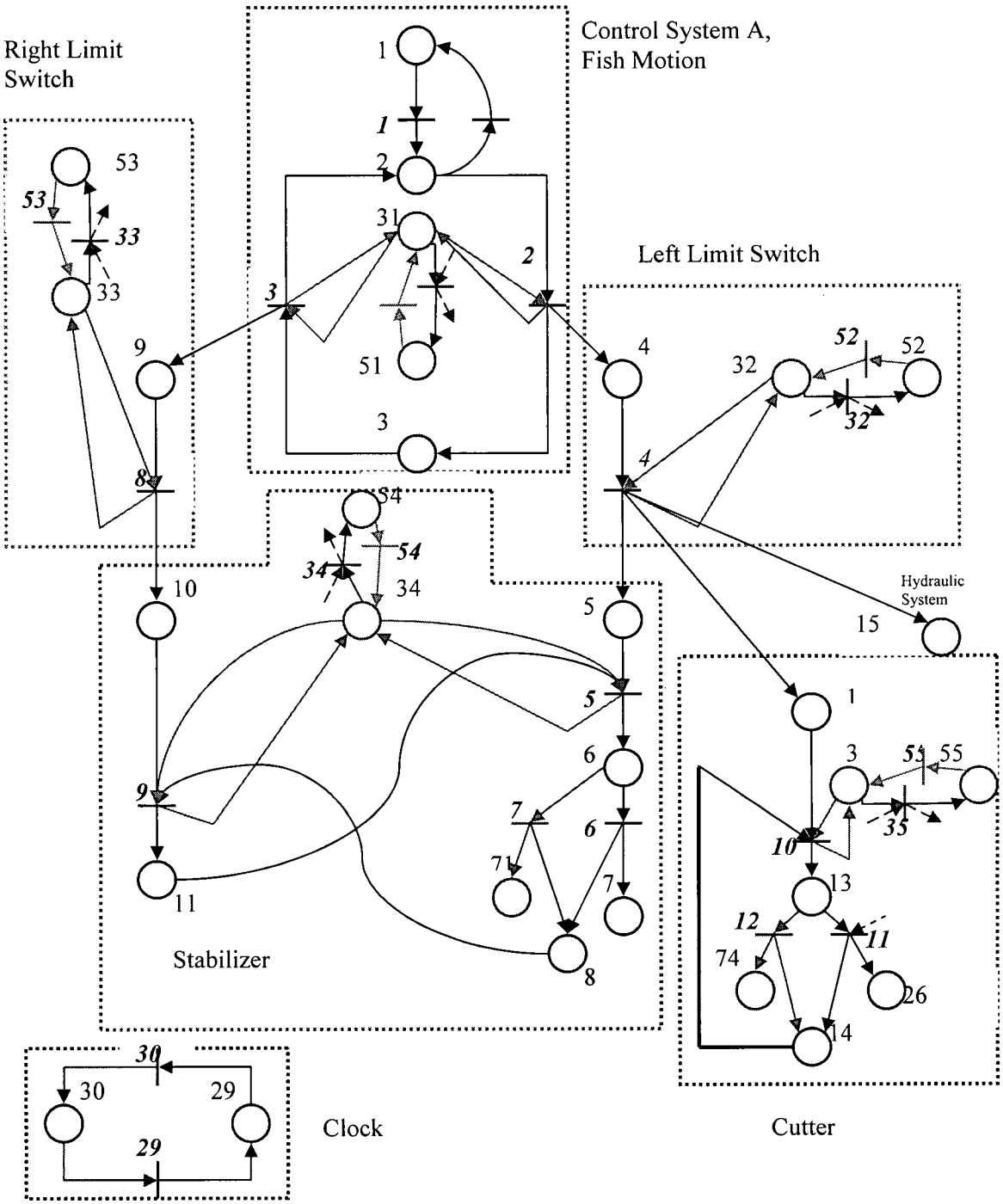


Figure 7.5: Petri-net model of the Iron Butcher.

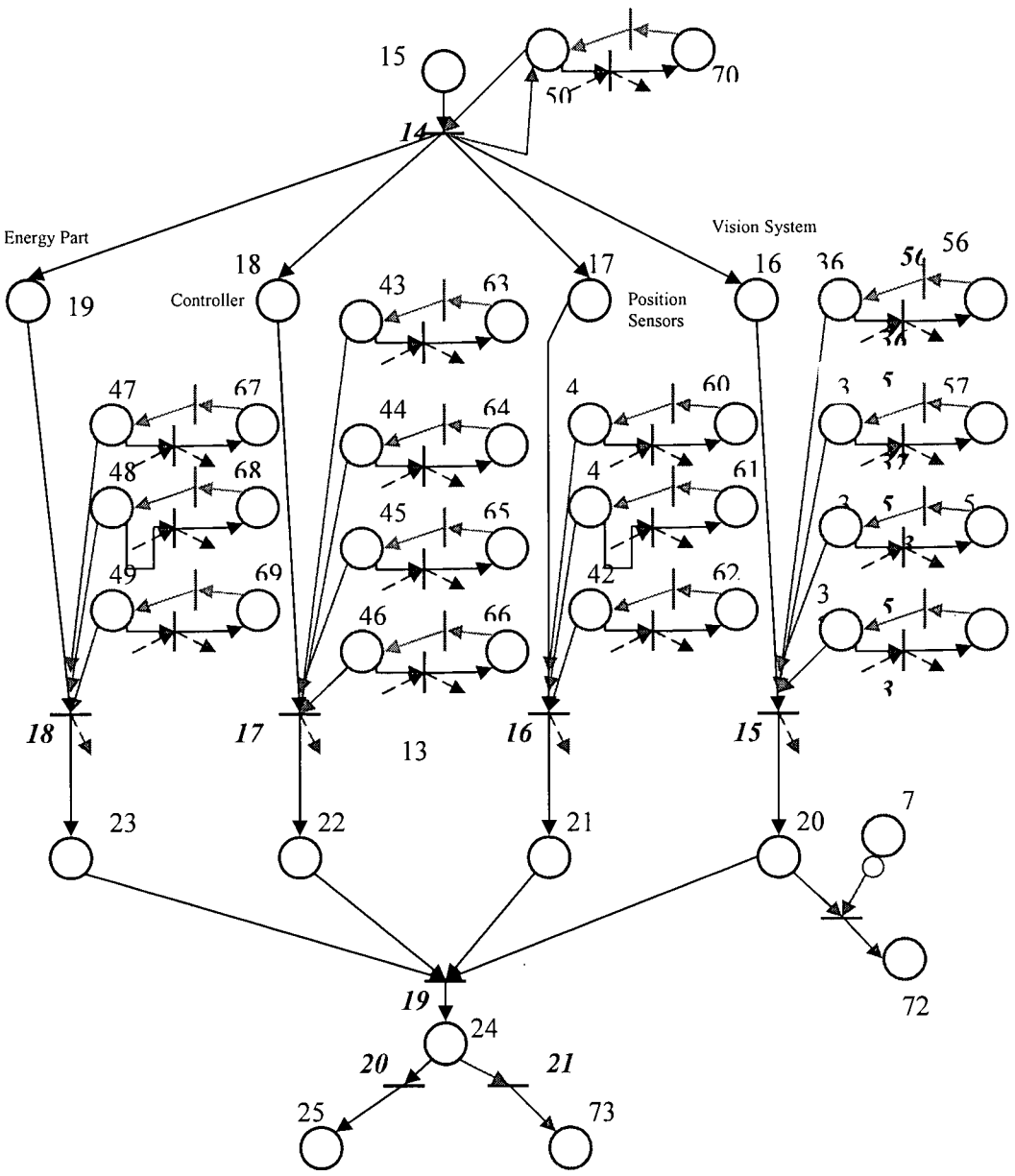


Figure 7.6: Petri-net model of the electro-hydraulic manipulator.

Table 7.6: Description of conditions (places) of Petri net model of the Iron Butcher.

Place	Description	Place	Description
1	Off	41	Sensors are calibrated
2	Push	42	Sensors are mounted properly
3	Pull	43	Controller software is OK
4	End-Push	44	Controller is tuned
5	Stabilizer is activate	45	Data acquisition card is OK
6	Stabilizer moves down	46	Amplifier is OK
7	Good stabilization	47	Servovalves are OK
8	Stabilizer is down	48	Pumping system is OK
9	End-Pull	49	Table is OK
10	Stabilizer is activate	50	Computer is OK
11	Stabilizer is up	51	Fish motion system is failed
12	Cutter is activate	52	Left switch is failed
13	Cutter is down	53	Right Switch is failed
14	Cutter is up	54	Stabilizer system is failed
15	Positioning system is active	55	Cutter is failed
16	Vision system is triggered	56	Camera is failed
17	Sensors are triggered	57	Camera is mounted properly
18	Controller is triggered	58	Frame grabber is failed
19	Energy parts are triggered	59	Software is failed
20	Vision system operation is done	60	Sensors are failed
21	Sensing is done	61	Sensors are calibrated
22	Controller job is done	62	Sensors are mounted properly
23	Energy part is done	63	Controller software is failed
24	Positioning is done	64	Controller is not tuned
25	Good positioning	65	Data acquisition card is failed
26	Cutting is done successfully	66	Amplifier is failed
27		67	Servovalves are failed
28		68	Pumping system is failed
29	Clock	69	Table is failed
30	Clock	70	Computer is failed
31	Fish motion system is OK	71	Stabilization is not suitable
32	Left switch is OK	72	Not proper image (taking photo before stabilization)
33	Right Switch is OK	73	Bad positioning
34	Stabilizer system is OK		
35	Cutter is OK		
36	Camera is OK		
37	Camera is mounted properly		
38	Frame grabber is OK		
39	Software is OK		
40	Sensors are OK		

7.3.1 Simulation Results

The Petri-net model of the Iron Butcher is used to assess the reliability specifications of this machine. To this end, 100 copies of the machine are simulated for a period of 30 years and their failure history is analyzed to assess the reliability specifications versus time. The average time to first failure is found to be 1093.9 days with a standard deviation of 1063 days. The general trend of TBF (Figure 7.7) shows a reduction in the time between failures indicating an aging effect.

Figure 7.8 shows the corresponding reliability, based on the severity of the failure modes. After 30 years, the reliability of this machine is found to be nearly 10%. This does not mean, however, that 90% of the machines are failed and are not operational. It indicates that at this time, copies of this machine would have experienced a high degree of failure, thereby causing a level of customer dissatisfaction which is equivalent to that of complete failure of 90% of the machines.

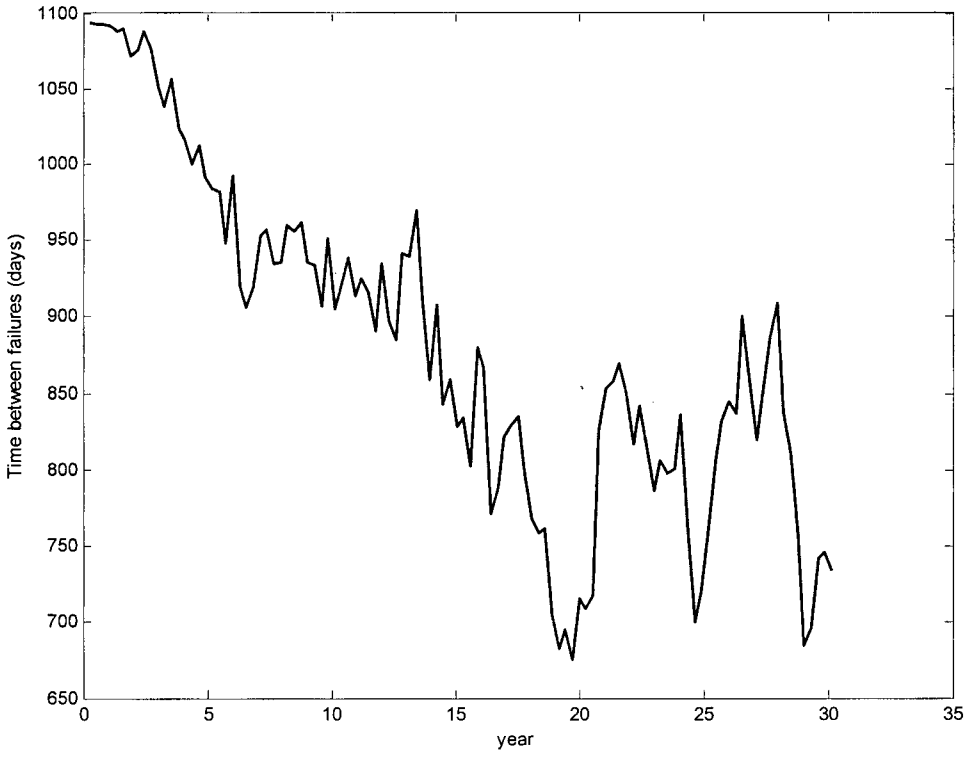


Figure 7.7: TBF distribution resulted from Petri-net simulation of the Iron Butcher.

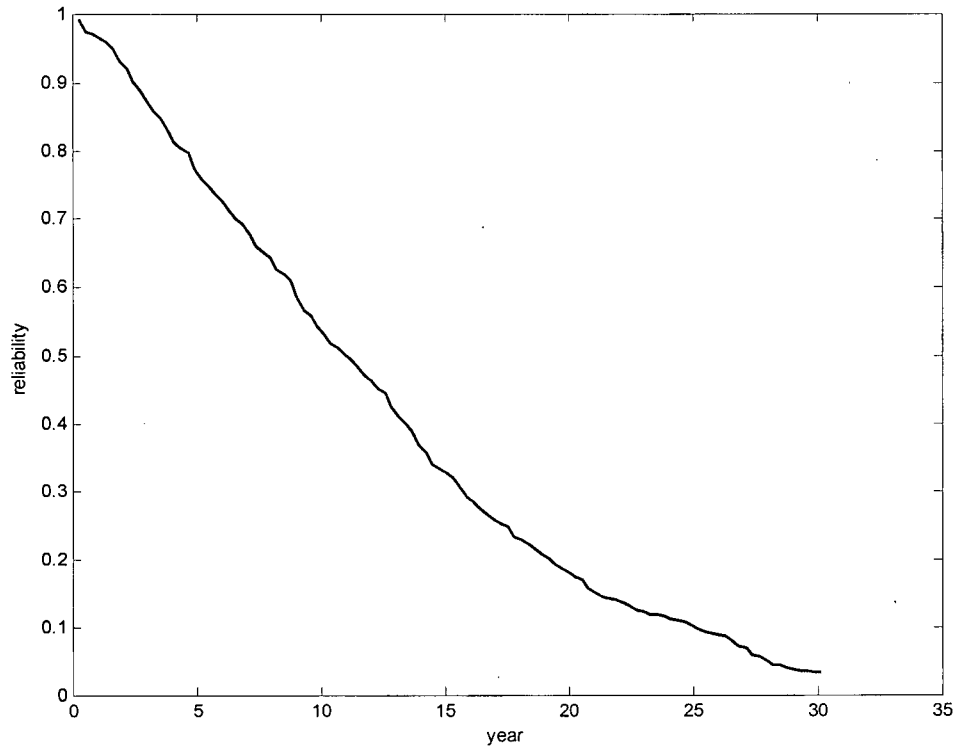


Figure 7.8: Equivalent reliability curve resulted from Petri-net simulation of the Iron Butcher.

7.3.2 Effect of Controller on Reliability

Better controller design can improve the reliability of the machine. The effect of the controller on system reliability may be investigated under three categories:

1. Self-tuning controller
2. Controller robustness
3. Reducing the failure probability of sensors and actuators by reducing the load and load fluctuations.

Possible failures which may be affected by the controller design are software failure and tuning failure. For example, by designing an intelligent supervisory controller for tuning the low-level direct controller, the tuning failure of the system can be resolved (De Silva, 1995; De Silva and Wickramarachchi, 1998a; De Silva and Wickramarachchi, 1998b).

Software failure is directly related to the controller design. In the case of model-based control, the software failure probability would be greater, especially if there is high uncertainty in the model. The software failure probability can be reduced by designing a robust and intelligent

controller (De Silva, 1995). Addition of a high-level supervisory controller can provide both self-tuning and robustness for the controller and therefore improve the reliability of the system. Figures 7.9 and 7.10 show the reliability improvement by adding a high-level supervisory control (dashed curves).

Another way in which the effect of the controller can be studied is through its effect on the failure rate of other components such as sensors and actuators. The controller sends a command to the actuator to control the system. If this command is not steady, then the actuator needs to change the load frequently, which would lead to a higher failure probability. In practice, a fluctuating actuation causes undesirable effects such as vibration, noise, wear, and fatigue (Li *et al.,* 2001). All these events have an inverse effect on the life of the components. The vibration can also cause mounting problems in the system (De Silva, 2006). For example, if the Iron Butcher produces high vibrations, the quality of the vision system can degrade. Other damaging effects of vibration include loosening of the camera, loosening of the sensors, problems in communication cables, and so on. It will also cause energy wastage in the system and can result in thermal problems.

Noise has an inverse effect on human reliability. It can also be a source of error in signal processing. Wear and fatigue as well degrade the quality of instruments and have an inverse effect on their life.

In the Weibull lifetime model, all these factors can be considered by changing α and β. In the presence of high fluctuations, α can be decreased, indicating a lower characteristic life. Similarly, β can be changed to represent the change in the failure rate pattern. The higher the β, the more severe the failure rate. It follows that controller design has a significant impact on the reliability of the system.

Figures 7.9 and 7.10 show the effect of lower controller command fluctuation on the reliability of the machine (dotted curves). Better reliability specifications can be achieved by combining both supervisory control and low command fluctuation features (dash-dot curves).

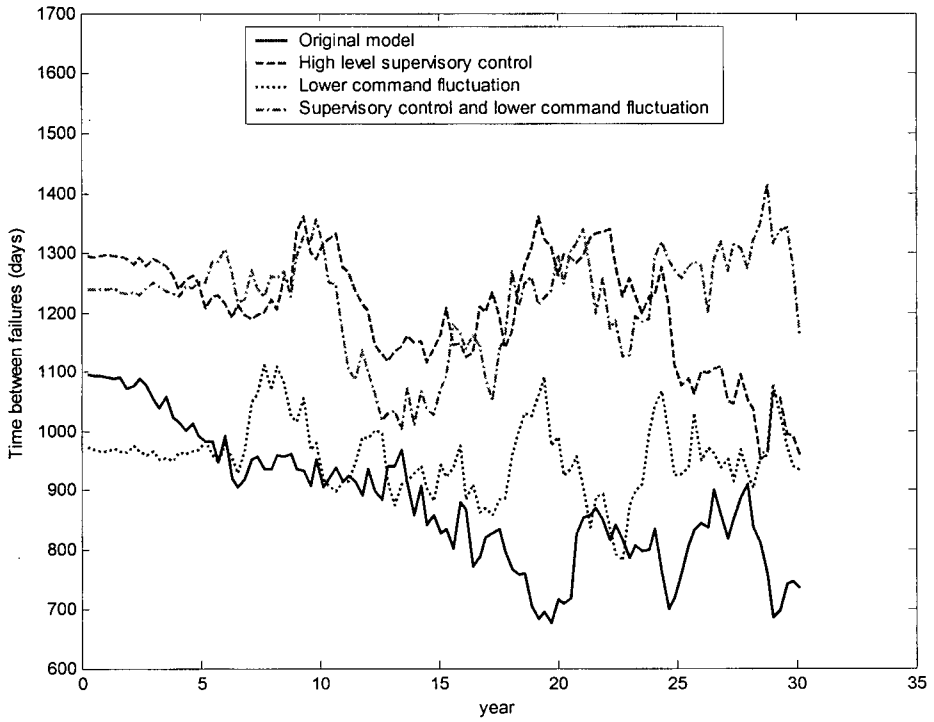


Figure 7.9: Improvement in TBF distribution of the Iron Butcher through controller design.

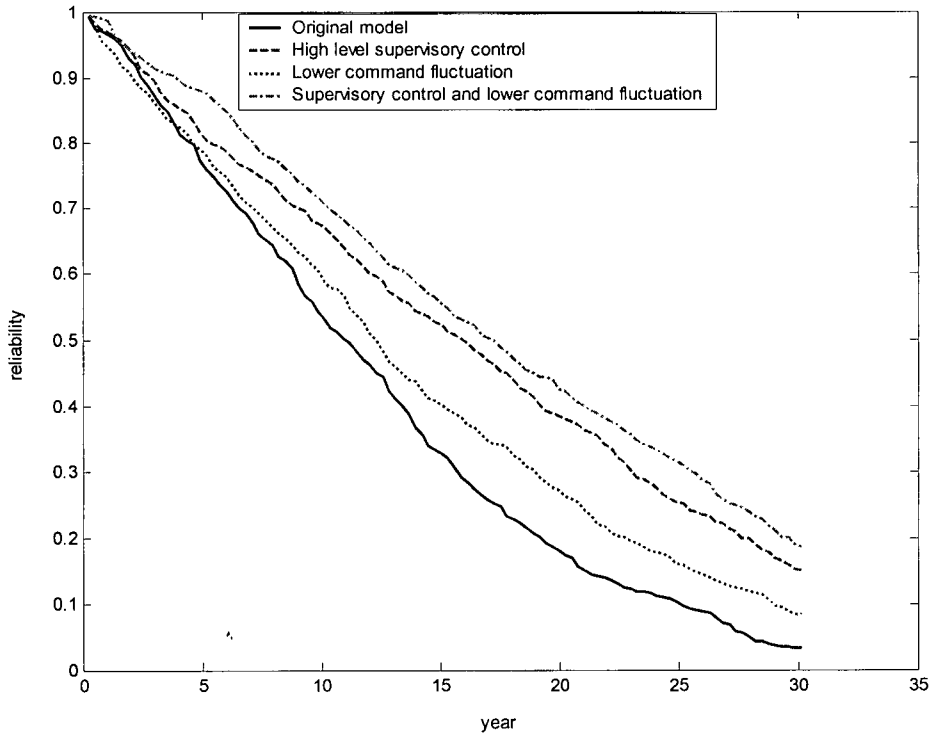


Figure 7.10: Improvement in equivalent reliability curve of the Iron Butcher through the controller design.

7.3.3 Redundancy

There are two types of redundancy:

1- Active redundancy

2- Passive redundancy

In active redundancy, the redundant components function simultaneously, even if they are not strictly necessary. If for example, there is an active redundancy with two components, it means that they both are in normal operating conditions of the machine, and if one component fails the other one will continue in normal operation without producing a noticeable effect on the output (Kaufmann *et al.,* 1977).

In Passive redundancy, a redundant component is reserved for use in place of the original component. If the original component fails, then the redundant one is put into service manually (Kaufmann *et al.,* 1977).

From the viewpoint of the reliability assessment approach developed in this chapter, the passive redundancy only affects the severity of a failure mode by reducing the repair time. It does not have any dynamic effect on the Petri-net model of the system. The assessment of active redundancy is somewhat more complicated. There are two ways to analysis active redundancy:

1- Redundant set as a subsystem: The set of redundant components can be modeled and assessed in isolation from other parts and as a sub-model. The results of the reliability assessment of the set can then be incorporated into the main modal. This approach, however, is not perfect because some internal events in the redundant set may not be taken into account. For example, if there are 10 numbers of a component and 6 of them are enough to perform the job, then the failure of the subsystem corresponds to the simultaneous failure of 5 of the redundant components. The repair time also corresponds to the required time for repair of the 5 failed components. A possible event that will be missed by this approach is the failure of one of the redundant components and its repair before the failure of any other ones.

2- Redundant components in the main modal: A more accurate but more complicated approach is to treat the redundant components similar to other components, model all of them in the main model, and show all the real operating conditions and repair policies in the main model. This makes the model more complicated but is necessary if the failure occurrence of each individual component also must be taken into account.

To improve the reliability of the Iron Butcher, an active redundancy is investigated for the pumping system. As the pumping system is rather costly, it may seem unreasonable to include a redundant unit in the machine. A more appropriate plan would be to share a pumping system

among several machines. It was found that the pumping system of the present machine is over-designed and its flow rate capacity is nearly double the required flow rate. If for example, n pumps are put in parallel to each other to provide the required flow rate for n machines, then, $n/2$ of the pumps are adequate to operate the machines, and $n/2$ pumps become redundant. It means that the machines will face a pumping failure only if $n/2+1$ out of n of the pumps fail simultaneously. As a pump system would be repaired as soon as a failure is detected, the probability of having $n/2+1$ failed pumps is rather low, particularly if n is high. Figure 7.11 shows the typical improvement in the pump system reliability by increasing the number of redundant pumps. For the case of sharing ten pumping systems between ten machines where five pumping systems are redundant, the failure probability of the pumping system is negligible. The resulting reliability improvement of the machine is evident from figures 7.12 and 7.13.

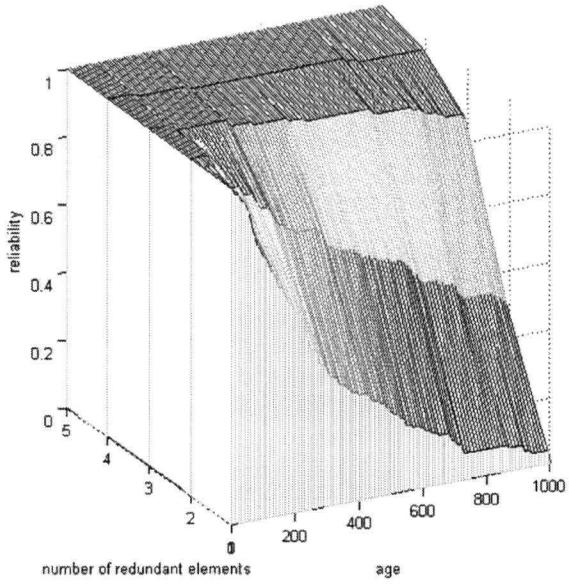


Figure 7.11: Reliability improvement in the hydraulic pump system through active redundancy.

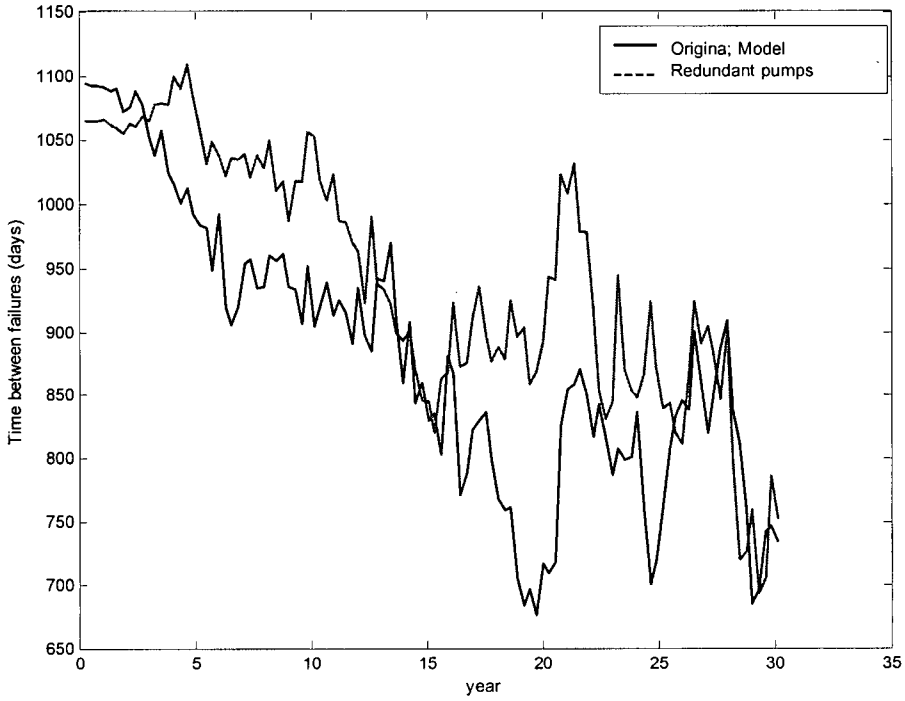


Figure 7.12: Improvement in TBF distribution of the Iron Butcher through active redundancy from shared hydraulic pumps.

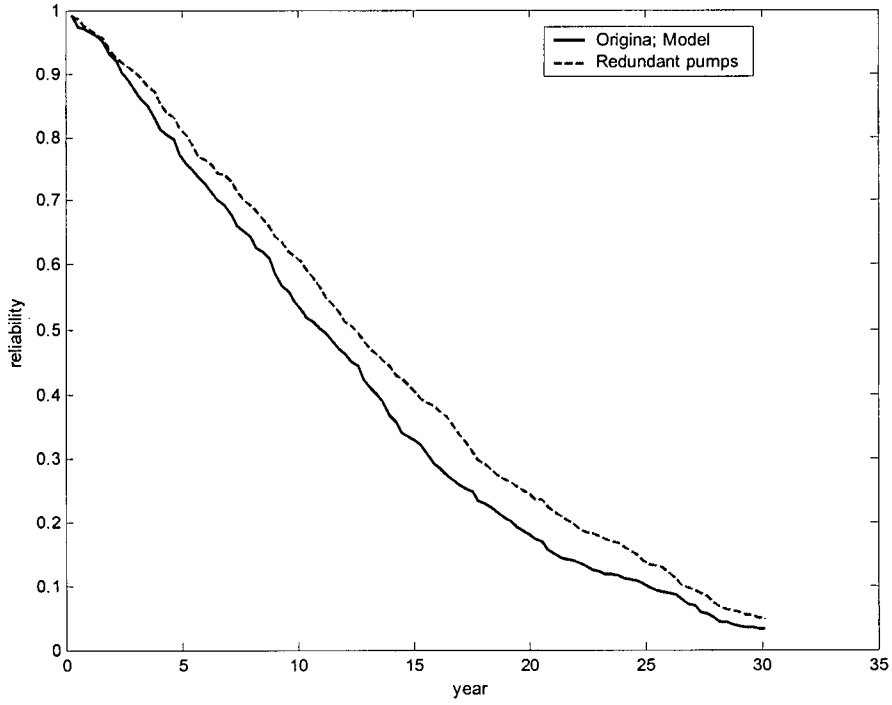


Figure 7.13: Improvement in equivalent reliability curve of the Iron Butcher through active redundancy from shared hydraulic pumps.

7.3.4 Fault Detection and Diagnosis

If a fault detection and diagnosis scheme is incorporated into a machine, it results in an increased sense of reliability in the customer. In the present reliability measurement approach, the effect of fault detection and diagnosis can be found in the severity assessment of a failure mode. Incorporating fault detection and diagnosis will facilitate the repair process and will reduce the severity of failure. In particular, partial scores for safety, detection and diagnosis, repair time, and repair cost will reduce and therefore, the reliability will improve.

7.4 Summary

Reliability is an important specification for evaluation of a mechatronic system. Dynamic architecture and severity assessment are two shortcomings of traditional reliability techniques, which were addressed in the developed approach for reliability analysis. A Petri-net simulation was developed as a powerful method to simulate what failure might happen to a machine in actual operation. The concept of failure mode severity was integrated into the Petri-net approach to provide a reliability specification for repairable machines and to make it as close as possible to the actual sense of reliability in the customer. Choquet fuzzy integral was incorporated and demonstrated as an axiomatic aggregation tool to reflect interactions between different criteria that contribute to the severity of a failure mode. The developed technique was successfully applied for a rather complex industrial mechatronic system. Some important issues in reliability assessment such as the effect of the controller, redundancy, and fault detection and diagnosis were investigated by this method.

Chapter 8
Conclusion

This thesis presented the development of new methodologies for evaluation and design of complex and multi-domain mechatronic systems. For a typical mechatronic system, where there exist an integration of several engineering domains such as mechanical, electrical, computer hardware, computer software, and control engineering, the interactions between subsystems and presence of multiple criteria require concurrent, integrated and system-based approaches for design and evaluation. This thesis investigated both theoretical and practical aspects of the optimal mechatronic design and developed several unified tools for the evaluation and optimal design of mechatronic systems. The contributions made in this thesis are outlined and their significance is indicated in the next section. The subsequent section presents some suggestions for further research in the area of mechatronic design.

8.1 Primary Contributions

In analytical aspects, this thesis made an important contribution by developing a design formulation, which demonstrated that a *System-based* thinking attitude was required in evaluation and design of mechatronic systems. Earlier work had proved that a mechatronic design should be integrated and concurrent in view of interactions between different components in a mechatronic system. The work presented in this thesis showed that since there are different interactive criteria in a realistic evaluation of mechatronic systems, the mechatronic design approach requires a system-based viewpoint.

The concept of mechatronic design quotient (MDQ) was revisited and its application on evaluation of complex mechatronic systems was formulated. The MDQ is used as a multi-criteria evaluation index reflecting general satisfaction score of a mechatronic design. Soft computing methods were applied for computing the MDQ for a complex system, taking into account the interactions between criteria and reflecting expert experience on mixed systems.

The MDQ provides a realistic multi-criteria evaluation of a mechatronic design; hence, it facilitates the development of a concurrent, integrated and system-based mechatronic design methodology. In the thesis, the MDQ was particularly utilized for evaluation of different choices of a mechatronic system in early stages of design. A systematic procedure was developed for optimal decision making in the conceptual mechatronic design.

For detailed design of a mechatronic system, a concurrent, integrated and system-based methodology was developed based on the concept of MDQ maximization. In this systematic procedure, a niching genetic algorithm was utilized with the aim to find an optimal representation for different possible configurations, with respect to some essential MDQ attributes – those which have veto effect on the MDQ evaluation. Subsequently a full and comprehensive MDQ evaluation was carried out to find the best solution among representations of different configurations.

Reliability is an important issue in optimum decision making about the configuration, hierarchical structure and operation policies of a mechatronic system. A new approach was developed for reliability assessment of repairable mechatronic systems. This approach possesses two main advantages over conventional methods. First, it can precisely simulate dynamic interactions and operational policies of a mechatronic system. Generalized stochastic Petri-net (GSPN) is used to mimic the real operation of a large number of units of a machine. Second, it considers the severity of failure modes in the reliability analysis. The Choquet integral is used to aggregate different severity attributes and calculate a severity index for each failure mode. This reliability assessment approach contributes in two stages of the developed design methodology. First is in the detailed design stage, once the designer intends to make a comparison between elite designs found by the niching genetic algorithm and selects the best. Second is in the final design stage, when the designer investigates different improvement possibilities and operational policies and has to make a tradeoff between different criteria in order to decide whether a particular improvement can be justified.

The extension of the integration of bond graph modeling and genetic programming for nonlinear mechatronic systems is another important contribution of this thesis. The outcome of this synergic integration is a unified evolutionary mechatronic tool. This tool can optimize both topology and "size" of a bond graph model to find the best solution for a problem. It can be used in any mechatronic problem, provided that an appropriate procedure for fitness evaluation is established for that problem. An original application of this unified tool for automated system identification of a complex mechatronic system was presented in the thesis, where both topology and size of the model were optimized.

8.2 Suggestions for Future Research

Fusion of multiple design criteria into one evaluation index is the key issue in MDQ evaluation and in the mechatronic design approach developed in this thesis. There is room for research in investigating different aggregation methods. Choquet integral is well-suited for this particular application, but a flaw is the exponential growth of the number of fuzzy measures to be specified corresponding to the number of criteria. There is room for research on methodologies to facilitate proper specification of these fuzzy measures. One possible approach is to use learning. The attitudes of costumers and experts can be gathered trough questionnaires. Acquired information from questionnaires can then be used in a learning procedure to specify fuzzy measures with the aim to appropriately represent the attitudes of costumers and experts.

As a learning scheme of Choquet fuzzy measures, one possible approach would be to use genetic algorithms. Evolutionary algorithms can be employed to find a set of fuzzy measures which has the best representation of acquired data from questionnaires.

In the present work, Petri-net modeling was used for the reliability analysis of complex mechatronic systems. Petri-net simulation can reveal some important information about the performance of a system. There is opportunity to use the Petri-net method for the evaluation of other criteria related to the functionality of complex systems, such as their flexibility and intelligence. In particular, for intelligence evaluation of a machine, the amount of human workload can be evaluated by simulating the real operation of a large number of that machine using generalized stochastic Petri-nets (GSPN). This can contribute in making proper decisions about the structure and operational policies so as to make the machine look more intelligent.

Another area which has the potential for further research is the integration of genetic programming and bond graph modeling. This synergic integration, which was extended for nonlinear systems in this thesis, can be utilized for automatic optimal controller design, where both topology and size of the controller are subject of optimization. For a nonlinear controller, such a tool can be particularly effective.

Finally, in this thesis the flexibility of the mechanical structure of the machine was not considered as a major issue. There is room for research on efficient integration of finite element analysis in the design methodology which has been developed in the thesis.

Bibliography

Altshuller, G.S., *Creativity as an Exact Science*, Gordon & Breach Science Publishing House, New York, 1984.

Amerongen, J.V., "Mechatronic design," *Mechatronics*, Vol. 13, No. 10, pp. 1045-1066, 2003.

Avigad, G., A. Moshaiov, and N. Brauner, "Toward a general tool for mechatronic design," *Proc. of IEEE Conference on Control Applications,* Vol. 2, pp. 1035-1040, 2003.

Behbahani, S., and C.W. de Silva, "Use of mechatronic design quotient in multi-criteria design," *Proceedings of International Symposium on Collaborative Research in Applied Science (ISOCRIAS)*, Vancouver, Canada, pp. 214-221, 2005a.

Behbahani, S., and C.W. de Silva, " Identification of a Mechatronic Model Using an Integrated Bond-Graph and Genetic-Programming Approach," *Proceedings of International Symposium on Collaborative Research in Applied Science (ISOCRIAS)*, Vancouver, Canada, pp. 158-165, 2005b.

Behbahani, S., and C.W. de Silva, "Automated bond graph modeling of a nonlinear mechatronic system," *ASME Transactions on Dynamic Systems, Measurement and Control,* 2005c (Under review).

Behbahani, S., and C.W. de Silva, "Mechatronic Design Quotient (MDQ) as the Basis of a New Multi-Criteria Mechatronic Design Methodology," *IEEE/ASME Transactions on Mechatronics,* 2005d (Accepted for publication).

Behbahani, S., and C.W. de Silva, "A New Multi-Criteria Mechatronic Design Methodology Using Niching Genetic Algorithm," *Proceedings of IEEE World Congress on Evolutionary Computation,* Vancouver, Canada, pp. 1031-1036, 2006a.

Behbahani, S., and C.W. de Silva, "Use of the Mechatronic Design Quotient in Multi-Criteria Design," *MECHATRONIC SYSTEMS — Devices, Design, Control, Operation, and Monitoring,* CRC Press, Taylor & Francis, Boca Raton, FL, 2006b (Accepted for Publication).

Behbahani, S., and C.W. de Silva, "Mechatronic Modeling and Design," *MECHATRONIC SYSTEMS — Devices, Design, Control, Operation, and Monitoring,* CRC Press, Taylor & Francis, Boca Raton, FL, 2006c (Accepted for Publication).

Behbahani, S., and C.W. de Silva, "An Evolutionary Mechatronic Tool," *MECHATRONIC SYSTEMS — Devices, Design, Control, Operation, and Monitoring,* CRC Press, Taylor & Francis, Boca Raton, FL, 2006d (Accepted for Publication).

Behbahani, S., and C.W. de Silva, "Reliability Trade off of a Complex Mechatronic System in Early Design Stages," *International Journal of Manufacturing Research,* 2006e (Accepted for Publication).
Bien, Z., W.C. Bang, D.Y. Kim, and J.S. Han, "Machine intelligence quotient: its measurement and applications," *Fuzzy Sets and Systems*, Vol. 127, No. 1, pp. 3-16, 2002.

Blischke, W.R., and D.N. Prabhakar Murthy, *Case Studies in Reliability and Maintenance,* John Wiley & Sons Inc., Hoboken, NJ, 2003.

Borutzky, W., "Bond graph modeling from an object oriented point of view," *Simulation Practice and Theory,* Vol. 7, pp. 439-461, 1999.

Borutzky, W., B. Barnard, and J.U. Thoma, "Describing bond graph models of hydraulic components in Modelica," *Mathematics and Computers in Simulation,* Vol. 53, pp. 381-387, 2000.

Borutzky, W., "Bond graph and object-oriented modeling – a comparison," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, Vol. 216, No. 1, pp. 21-33, 2002a.

Borutzky, W., and J. Granda "Bond graph based frequency domain sensitivity analysis of multidisciplinary systems," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, Vol. 216, No. 1, pp. 85-99, 2002b.

Borutzky, W., B. Barnard, and J. Thoma, "An orifice flow model for laminar and turbulent conditions," *Simulation Modeling Practice and Theory,* Vol. 10, pp. 141-152, 2002c.

Calvo, T., G. Mayor, and R. Mesiar, *Aggregation Operators, New Trends and Applications,* Physica-verlag Heidelberg New York, NY, 2002.

Cassanelli, G., F. Fantini, G. Serra, and S. Sgatti, "Reliability in automotive electronics: a case study applied to diesel engine control," *Microelectronics Reliability,* Vol. 43, No. 9-11, pp. 1411-1416, 2003.

Cellier, F.E., and R.T. McBride, "Object-oriented modeling of complex physical systems using the Dymola bond graph library," Available at:
http://www.scs.org/scsarchive/getDoc.cfm?id=2002

Cho, D.H., H. K. Jung, and C.G. Lee, "Induction motor design for electric vehicle using a niching genetic algorithm," *IEEE Transaction on Industry Applications*, Vol. 37, No. 4, pp. 994-999, July/August 2001.

Chocron, O., and P. Bidaud, "Evolutionary algorithms for global design of locomotion systems," *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 1573-1578, 1999.

Coelingh, E., T.J.A. de Vries, and R. Koster, "Assessment of mechatronic system performance at an early design stage," *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No. 3, pp. 269-279,

2002.

De Silva, C.W., *Control Sensors and Actua*tors, Prentice Hall, Inc., Englewood Cliffs, NJ, 1989.

De Silva, C.W., *Intelligent Control, Fuzzy Logic Application*, CRC Press, Boca Raton, FL. 1995.

De Silva, C.W., and N. Wickramarachchi, "Knowledge-based supervisory control system of a fish processing workcell; Part I: system development," *Engineering Applications of Artificial Intelligence*, Vol. 11, No. 1, pp. 97-118, 1998a.

De Silva, C.W., and N. Wickramarachchi, "Knowledge-based supervisory control system of a fish processing workcell; Part II: Implementation and evaluation," *Engineering Applications of Artificial Intelligence*, Vol. 11, No. 1, pp. 119-134, 1998b.

De Silva, C.W., "Sensing and information acquisition for intelligent mechatronic systems," *Proceedings of the Symposium on Information Transition*, Chinese Academy of Science, Hefei, China, pp. 9-18, Nov. 2003.

De Silva, C.W., *Mechatronics—An Integrated Approach,* Taylor & Francis, CRC Press, Boca Raton, FL, 2005.

De Silva, C.W., *Vibration—Fundamentals and Practice*, 2nd Edition, Taylor & Francis, CRC Press, Boca Raton, FL, 2006.

Demmou, H., S. Khalfaoui, E. Guilhem, and R. Valette, "Critical scenarios derivation methodology for mechatronic systems," *Reliability Engineering and System Safety,* Vol. 84, No. 1, pp. 33-44, 2004.

Dutuit, Y., E. Chatelet, J. P. Signoret, and P. Thomas, "Dependability modeling and evaluation by using stochastic Petri nets: application to two test cases," *Reliability Engineering and System Safety,* Vol. 55, pp. 117-124, 1997.

Fan, Z., K. Seo, J. Hu, E. Goodman, and R. Rosenberg, "A novel evolutionary engineering design approach for mixed-domain systems," *Engineering Optimization*, Vol. 36, No. 2, 2004.

Gantovnik, V.B., C.M. Anderson-Cook, Z. Gürdal, and L.T. Watson, "A genetic algorithm with memory for mixed discrete-continuous design optimization," *Computers and Structures,* Vol. 81, pp. 2003-2009, 2003.

GP tutorial, Available at: http://www.geneticprogramming.com/Tutorial/

Grabisch, M., "The application of fuzzy integrals in multicriteria decision making" *European Journal of Operational Research*, Vol. 89, pp. 445-456, 1996.

Granda, J.J., "The Role of bond graph modeling and simulation in mechatronics systems," *Mechatronics*, Vol. 12, No. 9-10, pp.1271-1295, 2002.

Guérin, F., M. Barreau, J. Y. Morel, A. Mihalache, B. Dumon, and A. Todoskoff, "Reliability analysis for complex industrial real-time systems: application on an antilock system," *Proceedings of the second IEEE International Conference on Systems, Man and Cybernetics,* Vol. 7, Hammamet, Tunisia, 2002.

Himeno, M., and R. Himeno, "The niching method for obtaining global optima and local optima in multimodal functions," *Systems and Computers in Japan*, Vol. 34, No. 11, pp. 30-42, 2003.

Introduction to genetic algorithms, Available at: http://cgm.cs.mcgill.ca/~soss/cs644/projects/marko/introduction.html

Ireson, W.G., C.F. Coombs, Jr., and R.Y. Moss, *Hand Book of Reliability Engineering and Management*, McGraw-Hill, New York, NY, 1996.

Karnopp, D., D.L. Margolis, and R.C. Rosenberg, *System Dynamics: Modeling and Simulation of Mechatronic Systems*, New York, Wiley, 2000.

Karray, F. and de Silva, C.W., *Soft Computing and Intelligent Systems Design*, Addison-Wesley, New York, NY, 2004.

Kaufmann, A., D. Grouchko, and R. Cruon, *Mathematical Models for the Reliability of Systems*, Academic Press Inc. New York, NY, 1977.

Kim, J.K., D.H. Cho, H.K. Jung, and C.G. Lee, "Niching genetic algorithm adopting restricted competition selection combined with pattern search method," *IEEE Transaction on Magnetics*, Vol. 38, No. 2, pp. 1001-1004, 2002.

Kogsino, N., L.T. Watson, Z. Gürdal, R.T. Haftca, "Genetic algorithm with local improvement for composite laminate design," *Structure Optimization,* Vol. 7, No. 4, pp. 207-218, 1994.

Koza, J.R., F. H. Bennett, D. Andre, and M.A. Keane, *Genetic Programming III, Darwinian Invention and Problem Solving,* Morgan Kaufman Publication, San Francisco, California, 1999.

Koza, J.R., F.H. Bennett, D. Andre, and M. A. Keane, "Synthesis of topology and sizing of analog electrical circuits by means of genetic programming," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2, pp. 459-482, 2000.

Kuang, M.L., M. Fodor, D. Hrovat, and M. Tran, "Hydraulic brake system modeling and control for active control of vehicle dynamics," *Proc. Of the American Control Conference*, San Diego, CL., pp. 4538-4542, 1999.

Li, Q., W.J. Zhang, and L.Chen, "Design for Control - A concurrent engineering approach for mechatronic system design," *IEEE Transaction on Mechatronics*, Vol. 6, No. 2, pp. 161-169, 2001.

Lumkes, J.H., and F.J. Fronczak, "Design, simulation, and validation of a bond graph model nad controller to switch between pump and motor operation using four on/off valves with a hydraulic axial piston pump/motor," *Proc. Of the American Control Conference*, Chicago,

Illinois, pp. 3605-3609, 2000.

Marichal, J.L., "An axiomatic approach of the discrete Choquet integral as a tool to aggregate interacting criteria," *IEEE Transaction on Fuzzy Systems*, Vol. 8, No. 6, pp. 800-807, 2000.

Marichal, J.L., "An axiomatic approach of the discrete Sugeno integral as a tool to aggregate interacting criteria in a qualitative framework," *IEEE Transaction on Fuzzy Systems*, Vol. 9, No. 1, pp. 164-172, 2001.

Mollica, R., and K.Y. Toumi, "A nonlinear dynamic model of a monotube shock absorber," *Proc. Of the American Control Conference,* Albuquerque, New Mexico, pp 704-708, 1997.

Moncelet, G., S. Christensen, H. Demmou, M. Paludetto, and J. Porras, "Analyzing a mechatronic system with coloured Petri net," *Int. J. STTT*, Vol. 2, pp. 160-167, 1998.

Moscrop, J., C. Cook, and F. Naghdy, "An analysis of motor/load inertia mismatch in machine tool servo systems," *Proceedings of IFAC Conference on Mechatronic Systems*, Darmstadt, Germany, Vol. 1, pp. 229-234, 2000.

Moulianitis, V.C., N. A. Aspragathos, and A. J. Dentsoras, "A model for concept evaluation in design - an application to mechatronic design of robot gripper," *Mechatronics*, Vol. 14, No. 6, pp. 599-622, 2004.

Park, H.J., B.K. Kim, and K.Y. Lim, "Measuring the machine intelligence quotient (MIQ) of human-machine cooperative systems," *IEEE Transaction on Systems, Man. and Cybernetics*- Part A: *Systems and Humans*, Vol. 31, No.2, pp. 89-96, 2001.

Paynter, H.M., *Analysis and Design of Engineering Systems*, MIT Press, Cambridge, MA, 1961.

Peterson, J.L., *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

Pil, A.C., and H.H. Asada, "Integrated structure/control design of mechatronic systems using a recursive experimental optimization method," *IEEE Transaction on Mechatronics*, Vol. 1, No. 3, pp. 191-203, 1996.

Sagirli, A., M.E. Bogoclu, and V. E. Omurlu, "Modeling the dynamics and kinematics of a telescopic rotary crane by the bond graph method (Part I)," *Nonlinear Dynamics*, Vol. 33, pp. 337-351, 2003.

Seo, K., Z.n Fan, J. Hu, E.D. Goodman, and R.C. Rosenberg, "Toward a unified and automated design methodology for multi domain dynamic systems using bond graphs and genetic programming," *Mechatronics*, Vol. 13, No. 8-9, pp. 851-885, 2003.

Tafazoli, S., C. W. de Silva, and P. D. Lawrence, "Tracking control of an electrohydraulic manipulator in the presence of friction," *IEEE Transactions on Control Systems Technology*, Vol. 6 , No. 3 , pp. 401 – 411, 1998.

Van Brussel, H.M.J., "Mechatronics – A powerful concurrent engineering framework,"

IEEE/ASME Transaction on Mechatronics, Vol. 1, No. 2, pp. 127-136, 1996.

Van Brussel, H.M.J., P. Sas, I. Nemeth, P. Fonseca, and P.V. den Braembussche, "Toward a mechatronic compiler," *IEEE/ASME Transaction on Mechatronics*, Vol. 6, No. 1, pp. 90-105, 2001.

Viersma, T.J., *Analysis, Synthesis and Design of Hydraulic Servo Systems and Pipelines*; Elsevier Scientific Publishing Company, Amsterdam-Oxford-New York, 1980.

Volovoi, V., "Modeling of system reliability Petri nets with aging tokens," *Reliability Engineering and System Safety,* Vol. 84, pp. 149-161, 2004.

Wang, J., Z. Fan, J.P. Terpenny, and E.D. Goodman, "Knowledge interaction with genetic programming in mechatronic systems design using bond graphs," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 35, No. 2, pp. 172-182, 2005.

Zhang, W.J., Q. Li, and L. S. Guo, "Integrated design of mechanical structure and control algorithm for a programmable four-bar linkage," *IEEE/ASME Transaction on Mechatronics*, Vol. 4, No. 4, pp. 354-362, 1999.

Zhou, M., and K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems, a Petri Net Approach,* World Scientific, Singapore, 1999.

Appendix A

Petri Net Modeling

This appendix defines some basic concepts of Petri nets (PNs) and shows how this tool can be used to model the functional behavior of mechatronic systems.

In view of the complex nature of mechatronic systems and interaction between operations of subsystems, the design of architecture and operation of these systems require modeling and analysis in order to select the optimal design alternative and employ most effective operation and repair policies. Petri nets prove to be an effective tool to establish a dynamic model of the functional behavior, and they are often the only possible approach. Another alternative approach is the Markov approach; however, it quickly reaches its limits when the number of states increases.

Petri nets are a graphical and mathematical tool for the analysis of systems. They provide a uniform environment for modeling, analysis and design of discrete-event systems. Analysis of Petri nets can reveal important information about the structure and dynamic behavior of the modeled system. This information can be used to evaluate the modeled system and suggest improvements and changes, or to make proper selection from several available choices. By this, Petri net modeling can contribute in the design process of mechatronic systems.

In Petri-net modeling of a mechatronic system, the system is characterized by its dynamical architecture. Possible *events* and *conditions* in the real operation of the system are included in the Petri-net model, such as normal states of operation of the system and also possible states leading to critical feared states and states of repair. Different events are fired depending on the system architecture and possible states of the machine.

A.1 Basics of Petri Nets

The Petri nets represent a bottom-up method, which utilizes a symbolic language. A Petri net *Z=(P, T, I, O)* has four arguments, where:

- $P = \{p_1, p_2, p_3, ..., p_n\}$, $n>0$ is a finite set of *places* indicated by circles.
- $T = \{t_1, t_2, t_3, ..., t_s\}$, $s>0$ is a finite set of *transitions* indicated by lines.
- $I : P \times T \rightarrow \mathbb{N}$ is an input function which defines the set of directed arcs from P to T.
- $O : P \times T \rightarrow \mathbb{N}$ is an input function which defines the set of directed arcs from T to P.

$$P = \{p_1, p_2, p_3\}$$

$$T = \{t_1, t_2\}$$

$$I = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad O = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$m = (1,1,0)$$

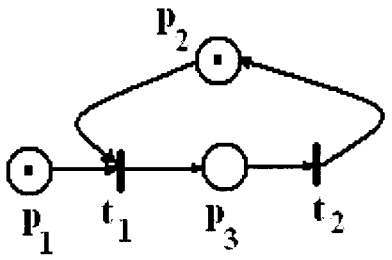


Figure A.1: An example of a Petri-net model and its mathematical representation.

A marked Petri net *Z=(P, T, I, O, m)* has one more argument:

- $m : P \rightarrow N$ is a marking whose i^{th} component represents the number of tokens (represented by dots) in the i^{th} place. An initial marking is denoted by m_0.

Figure A.1 shows a sample Petri net and its components.

A.2 How to Model a System by Petri Nets

A Petri-net graph allows the user to represent the actual operation of system functions and use markings to assign tokens to a net (Ireson *et al.*, 1996). Two primitive concepts exist on the modeling of a system by Petri nets: *events* and *conditions.* Events are the actions which take place in the system, either as normal operation of the system or events corresponding to the failure and unexpected operation of the system. In contrast, a condition is a predicate or logical description of the state of the system. They can be either conditions due to normal operation of the system or conditions caused by a failure or faulty scenario. For example, "failure of a component" is viewed as an event which causes the state of a component change form "being healthy" to "being failed." These two states are viewed as conditions in the Petri-net viewpoint of the functional behavior of a system. The occurrence of an event may need certain conditions. These are called *preconditions* of the event. The occurrence of the event may cause new conditions on the system which are called *post conditions* (Peterson, 1981).

This view of a system can be easily modeled as a Petri net. Conditions are modeled by *places* and events are modeled by *transitions*. The inputs of an event are preconditions and the outputs

are post-conditions. The occurrence of an event corresponds to the firing of the corresponding transition. The holding of a condition is represented by a token in the place corresponding to the condition. When a transition fires, it removes the enabling tokens representing the holding of preconditions, and creates new tokens representing the holding of post-conditions (Peterson, 1981).

There exist different classes of Petri nets. A stochastic Petri net (SPN) is obtained by associating a firing time with each transition. A generalized stochastic Petri net (GSPN) allows timed transition with zero firing times and exponentially distributed firing times. The extended stochastic Petri net (ESPN) allows the firing times to belong to an arbitrary statistical distribution (Ireson *et al.,* 1996).

A.3 Execution Rules

Execution rules include enabling and firing rules (Zhou *et al.,* 1999):

- A transition $t \in T$ is enabled if and only if

$m(p) \geq I(p,t), \quad \forall p \in P$.

- Marking m, firing t results a new marking m':

$m'(p) = m(p) - I(p,t) + O(p,t), \forall p \in P$

One significant extension to increasing the modeling power of Petri nets is achieved by adding the zero testing ability, i.e., the ability to test whether a place has no token (Zhou *et al.,* 1999). An *inhibitor* arc is used to implement this ability. The inhibitor arc connects an input place to a transition and is pictured by an arc terminated with a small circle. A small circle denotes a "not" in digital circuits.

It can formally be introduced into the PN definition by adding an inhibitor function $H : P \times T \rightarrow \mathbb{N}$. A transition is regarded as enabled if and only if:

$m(p) \geq I(p,t), m(p) < H(p,t), \forall p \in P$

A.4 Analysis of Petri nets

There are some common techniques to assess a system by its Petri-net model. Some important concepts are listed below which can be further studied in the reference by Zhou *et al.* (1999).

1- Reachability

2- Boundedness and safeness

3- Conservativeness

4- Liveness

5- Reversability and home state

These analysis techniques represent some important properties of the system. In addition to them, siphon and trap-based analysis methods may be used to analyze some special classes of Petri nets (Zhou *et al.,* 1999). Its usage in certain subclasses of Petri nets can avoid the state explosion problems encountered in using the reachability analysis method.

A Petri-net model can be used to derive critical feared scenarios of a system and to create a fault tree (Demmou *et al.,* 2004).

A.5 Simulation of Petri Nets

For complex Petri-net models, discrete-event simulation is an effective way to check the system properties. The idea is simple, specifically, using the execution algorithm to run the net and mimic the operation of the system. This simulation can be an expensive and time-consuming technique. It can reveal the presence of undesirable properties but cannot prove the correctness of the model in general cases. Despite this, Petri-net simulation is indeed a convenient, straightforward, and effective approach for engineers to validate the desired properties of a complex system.

A simulation procedure based on Petri nets is outlined below (Zhou, *et al.* 1999):

1. Initialization: decide the initial marking and the set of enabled transitions at the marking.

2. If the number of preset simulation steps or certain stopping criteria are met, stop simulation; otherwise, if there is no transition enabled, report a dead marking and either stop or go to step 1.

3. Randomly pick a transition to fire. Remove the same number of tokens from each of the input places as the number of arcs from the place to the transition, and deposit the same number of tokens to each of its output places as the number of arcs from the transition to that place.

4. Remove enabled transitions that are modified at the new marking by checking the output transitions of the input places used in step 3. If the output transitions of the output places in step 3 become enabled, add those enabled ones. Go to step 2.

This algorithm can be modified to simulate extended Petri nets such as timed Petri nets. Two important modifications are:

1- Policies are needed to pick up a transition among conflicting ones.

2- The firing time delay needs to be determined, which may be deterministic or sampled from a given random time delay distribution.

The conflict resolution can be performed by knowing the probability of each transition. In the context of reliability analysis, it is necessary to check whether a component is healthy or not each time the component is fired to do its task. To solve this conflict, the failure rate or failure probability of that component is necessary. Then, one of the transitions is picked randomly, according to the probability of conflicts.

Simulation of timed Petri nets allows one to derive the temporal performance for a system under very realistic assumptions in addition to detection of the presence of the behavioral problems, in a cost effective manner. The performance measures include equipment utilization, system throughput, work process, failure occurrence, repair time, repair cost, and reliability.