The Dissertation Committee for Zhaohong Wu Certifies that this is the approved version of the following dissertation:

AUTOMATED OPTIMAL DESIGN OF DYNAMIC SYSTEMS

Committee:

Matthew I. Campbell, Supervisor

Benito R. Fernandez, Supervisor

Richard H. Crawford

Raul G. Longoria

Risto Miikkulainen

AUTOMATED OPTIMAL DESIGN OF DYNAMIC SYSTEMS

by

Zhaohong Wu, B.E.; M.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2007

Dedication

This dissertation is dedicated to the memories of my beloved grandfather, Fengzhi Wu, who passed away in the year 2000. He was an outstanding artist in Chinese calligraphy. His righteous and generous personality as also demonstrated in his artworks affected me and numerous others. He was an honorable man and a lovely grandfather.

Acknowledgements

I thank my delightful wife Yanjing, who support me, encourage me and look into our life with love and wisdom. Her quickly-developed cooking skills, braveness of giving birth to our daughter Kristin and excellence in her own MBA study bring me great strength to finish my dissertation study. This studying process we experienced will inspire us through our life path.

I would like to acknowledge a tremendous debt of gratitude towards my supervisors, Professor Benito R. Fernandez and Professor Matthew I. Campbell, whose patience, understanding, and personal guidance to me cannot be overstated. It was through numerous times of discussion with them to arrive at this point of completion that symbols the sparking in emerging two areas of expertise.

I am deeply grateful for the many supports from Professor Koen. He was the first professor I had class with at UT and was the professor I worked three years as his teaching assistant. I was impressed not only with his knowledge but also his vision and passion of education. Dr. Koen also paid very attention to possible opportunities that are beneficial to my future career.

Next, I want to take this opportunity to thank my committee members, Professor Richard H. Crawford, Professor Raul G. Longoria, and Professor Risto Miikkulainen for their time and diligence. Special mention goes to my friends Drs. Fu Zhang and Guohua Ma, who picked me up from the bus station on my day of joining UT and in the following years cared me, encouraged me and shared with me study experience in mechanical engineering. I would also like to thank Ms. Cindy Raman and Ms. Ruth Schwab, who keeps the graduate study ship afloat for the ME Department, and who helped navigate me through the graduate program. Further, I would like to thank the members of the Madlab and Nerdlab in UT Mechanical Engineering too numerous to mention by name, whose fun-loving research environment let me enjoy my study and life at UT.

I thank my parents, who not only made my life possible, but who also give me their unwavering moral support and guidance through thick and thin. I specially want to express my deep appreciation to my parents-in-law who trusted me by letting their beloved daughter marry a student in a foreign land across the Pacific, and love their daughter and son in law from their hearts. They together helped take care of our new-born daughter here and in China to let my wife and I successfully finish our study. I am ever grateful for them.

AUTOMATED OPTIMAL DESIGN OF DYNAMIC SYSTEMS

Publication No._____

Zhaohong Wu, Ph.D.

The University of Texas at Austin, 2007

Supervisors: Matthew I. Campbell and Benito R. Fernandez

The A-ODDS (Automated Optimal Design of Dynamic Systems) method proposed in this dissertation is generally represented by two iterative search processes (loops) linked by automated modeling of design topologies. The first loop is for topology generation and the second loop is for tuning the parameters for a design topology.

In the first loop, a design synthesis method is proposed that combines a probability based decision making strategy and design grammars (production rules) into a "design agent" for system development. The decision making strategy can be evolved to facilitate the exploration of a multi-domain design space in a topologically open-ended manner, and still efficiently find appropriate design configurations. Probability based decision making is applied at each stage of topology development. Experimental results show that a design agent demonstrates steady performance in terms of the overall fitness

of the designs it generated. A good design strategy or agent has a better chance of producing superior designs.

The research in automated modeling and in the following second tuning loop is leading to a computer-aided design tool in which engineering designers can test various design concepts (topologies) in an environment equipped to automatically model the dynamics and conveniently optimize the specified components, given the evaluation criteria pre-defined by human designers. Automated modeling of design configurations is facilitated by a design representation named as CD-Graph (Conceptual Dynamics Graph) and generic models of various components. CD Graphs record the coupling formats in which not only physical components, but also their generic models are assembled topologically in the same fashion. A generic component model can accommodate various types of coupling between this component and its environment. In the second loop a systematic approach is proposed to automatically prepare a design problem for the convenient application of parameter optimization. This preparation encodes and decodes proper design variables into design genotypes, while taking into consideration of the design constraints and physical constitutive laws.

Table of Contents

List of Tables

List of Figures

Chapter 1 INTRODUCTION

In the last few decades, computers have given rise to fundamental changes to engineering design practice. Various computational tools, such as visualizing systems (Computer Aided Design) and analyzing systems (Finite Element Analysis), are used for design and testing prior to constructing prototypes. Such computational convenience can reduce design costs and cycle times by predicting difficulties early in the design process.

In spite of these advances, the role of the computer in designing new artifacts or even individual components has been limited. Knowledge of components and design process has yet to be leveraged in a systematic manner towards automated design. Given the vast number of independent or OEM (original equipment manufacturers) components used in many of today's technological artifacts and the quick cycle times imposed in the creation of new devices, it is desirable to explore the capability of computers for combining automated topology synthesis and topology evaluation. Automatic synthesis of topology has proven to be much more challenging than the simple parametric optimization of a pre-existing component arrangement since it is a challenge that needs effective and efficient support of topology evaluation.

The evaluation task can be conducted experimentally or computationally. Experimental evaluations test a set of physical parameters to give realistic outputs of the

system performance while usually taking more time and labor for limited number of experiments. Computational evaluations call for a system model to be built before simulations can be run against a set of parameters but very often it is not trivial to obtain the system model automatically. After a model is obtained, optimization can be used to effectively help determine the best values for parameters which also minimize the number of computational evaluations.

With the establishment of state-space search (originally formulated as a cognitive model of human problem solving by Newell and Simon, 1972; Simon, 1969), computational design has slowly progressed from solving well-behaved mathematical problems towards addressing more ambitious engineering design problems. The slower development of computational design can be attributed to three challenges unique to design:

- First, characteristics of the human design process (hereafter referred to as "human design") have yet to be realized in an implemented process. While research in automated design involves static problems, real design is not static. Human creativity is capable of developing novel artifacts through adapting to difficulties and challenging past conventions. One of this dissertation's goals is to establish a new theory for automated computational design that incorporates characteristics of human conceptual designing, thereby broadening the applicability of computers in engineering.
- Second, design requires the comparison of various alternatives. Such comparison is best performed by evaluating solutions on a common metric. Depending on the complexity of the design problem, this comparison may require detailed analysis

to occur as a subset of design. However due to the numerous design alternatives being compared, automated analysis must be structured to allow for a quick evaluation of alternatives. The surviving candidates can then be applied detailed evaluations. As a result of the time constraints, the computational design researcher may need to develop both heuristics and complex analysis to address the challenge of both searching for and automatically analyzing designs in real time. One main effort of this dissertation is to develop a framework for automated modeling and optimization to support design concept evaluation qualitatively and quantitatively.

- Third, engineering analysis has a history of mathematical formalism at its foundation; however, engineering design has yet to be studied with the same rigor. When a design topology is not producing good results or more specifically the current value setting of the parameters is not giving satisfactory performance, the searching process needs guidance on how and where to move next. Various optimizations use gradient–based approach or biological inspired approach to guide the searching process. Search guiding becomes specifically meaningful when experimental approach is to be used for evaluation of design concepts with limited time and cost projection.

This dissertation work establishes a method for Automated Optimal Design of Dynamic Systems (A-ODDS). A-ODDS brings together various innovations in design theory, automated dynamic system modeling, and optimization to support the early phase of conceptualization that occurs only when a design need has been established.

1.1 MOTIVATION

Design is a major activity of practicing mechanical, electrical, civil, and aeronautical engineers. The design process entails creation of a complex structure to satisfy user-defined requirements. Since the design process typically entails tradeoffs between competing considerations, the end product of the process is usually a satisfactory and compliant design as opposed to a perfect design.

The conceptual design of dynamic engineering systems is one of the most demanding and yet poorly understood aspects of mechanical engineering. It requires the designer to assemble a multitude of parts selected from a larger discrete set, that contains physical properties and parameters that may have continuous or discrete values. At the same time, the designer must satisfy a variety of constraints (limits, inequalities, etc.) and reconcile conflicting cost functions. In a typical design scenario, a design problem will call for the engineers to meet specifications for various qualities of an electro-mechanical device such as constraints on power, bandwidth, accuracy, or stress failure. Mechatronic devices represent a confluence of efforts including mechanical, electrical, control, computer, and manufacturing engineers.

This doctoral research aims for an integrated framework that will both evolve new topologies of components, and optimize the parameters of those components. The target application is to aid the user in the design of complex electromechanical or mechatronic devices. Mechatronics is finding applications in a number of artifacts from home appliances, to space-bound robots. However, the complexity of such systems requires countless design hours to create a robust product. The A-ODDS method proposed here

could eventually ease the burden of design by offloading design problems onto a computational system capable of searching a vast number of alternatives.

By automatically creating and optimizing topologies, the designer's duties are alleviated. This can be especially true in industries seeking to provide customized products to suit each individual's unique needs such as "massive customization" or "one of a kind" designs. The ability to appeal to an individual customer's needs is an ideal widespread in modern product design environments. Providing the "service" or special attention required of each individual while still offering products in high volumes has only been attainable from companies leveraging computational power and quick communication via the Internet. Examples of such products include customized shoes, customized bicycles, and customized computers.

Furthermore, one can also see benefits in offering customized products to those who may depend on a specific product. Assistive technology is aimed at providing tools to individuals who are unable to perform daily tasks that many take for granted. Disabled individuals each have a unique set of motor skills that should be leveraged by assistive technologies to allow them to enjoy the comforts of an independent lifestyle. However, such products are often developed to target a wide variety of skill sets. Research aimed at offloading typical design problems involving dynamic fixtures or mechatronic paraphernalia could lessen design times and produce higher quality products specifically tailored to the individual.

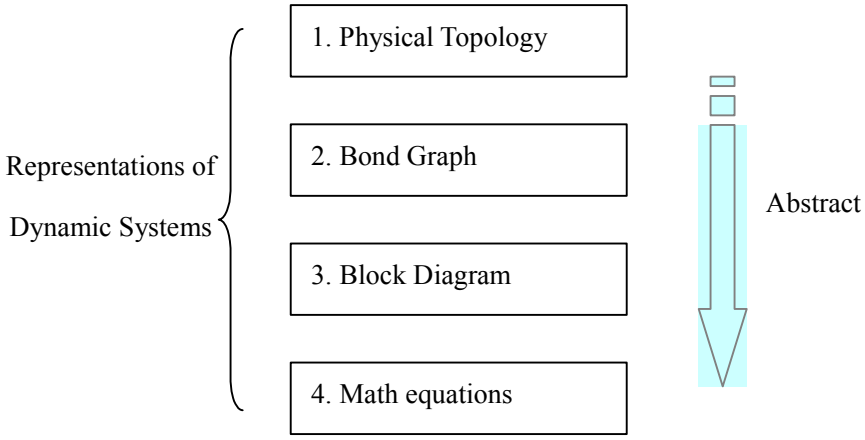
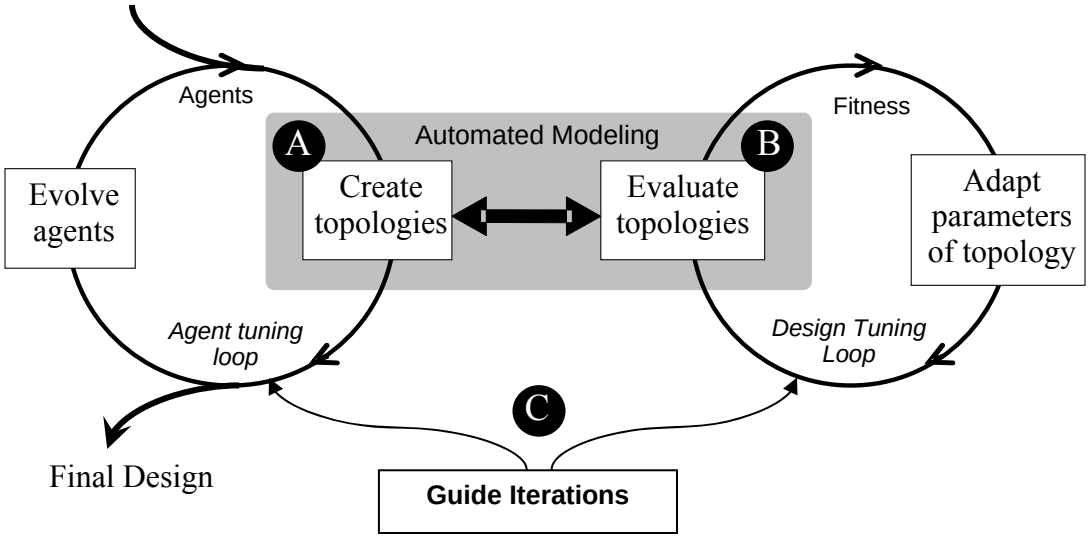


Figure 1-1: Dynamic system representations

1.2 COMPUTATIONAL DYNAMIC SYSTEM DESIGN

A dynamic system can be represented by a topology of physical component, or other abstract approaches such as bond graphs, transfer function, and mathematical equations in an order of increasing abstraction levels as shown in Figure 1-1. In this dissertation, although efforts are made to conduct conceptual dynamic system design at the bond graph level, significant work has been done to provide support for the physical topology level design in return using bond graphs as an approach for modeling these physical designs.

The A-ODDS (Automated Optimal Design of Dynamic Systems) method (Figure 1-2) proposed in this research is generally represented by two iterative search processes (loops) linked by automated modeling of design topologies. The left loop is for topology generation and the right loop is for tuning the parameters for a design topology. Specifically, three general research challenges are identified as followed (alphabetically marked in Figure 1-2).



General Research Challenges:
A – Design Topology Synthesis (Chapter 4)
B – Automated Modeling and Optimization Formulation (Chapter 5)
C – Iteration Guidance (Chapter 4&5)

Figure 1-2: Automated Optimal Design of Dynamic Systems (A-ODDS)

1.2.2 Creating Topologies

A challenge in computational design is the method for generating design concepts from a design problem description. The top of Figure 1-3 depicts the process of generating design states. The process starts with a "seed" or description of the design problem (as shown by the bolded circle at the top of the figure).

All design alternatives are constructed by design agents in stages progressing from problem description to complete design instance. Design agents are equipped with design grammar rules (e.g., allowing a gear attached to a shaft, but not to a spring) and probabilistic strategies (e.g., the probabilistic choice over a lever or a motor for the task

of energy transformation). Grammar rules can guarantee the effectiveness the designs generated and the probabilistic strategies can help to make decisions when two or more options exist. The result of the construction is a point in the search space shown in the center of Figure 1-3. If the design specification has not been reached, the design agents can be evolved (or updated) to produce next generation of designs with different attributes. Ideally, the generation method should be capable of creating the wide diversity of solutions.

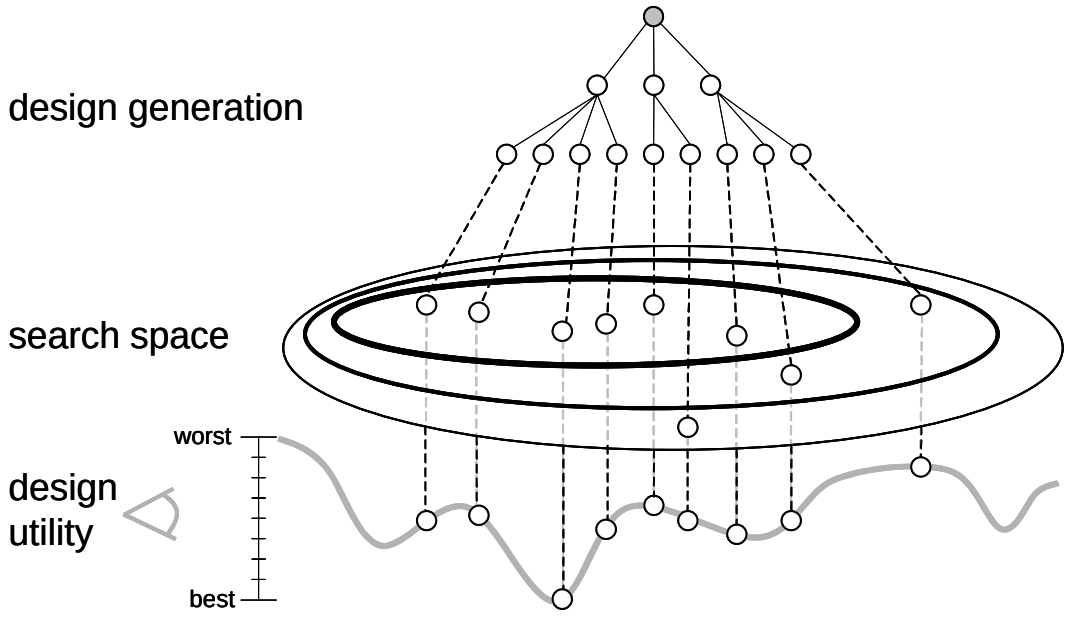


Figure 1-3: View of search space as an intermediate step between the process of creating design states and evaluating the utility of such states. [Campbell, 2000]

1.2.3 Evaluating Topology

In order to direct the search process, a computational system must have the ability to compare design states. While design is in essence an open-ended problem with "no

right answer", definite distinctions can be made between good and bad solutions. A metric may be constructed to articulate these distinctions as an overall value for individual design states. A numerical value, no matter how approximate, is determined for each unique design in the search process, as seen in Figure 1-3. In the bottom of the figure, the utility of each design is visualized as a numeric value on a surface of "evaluated" design states.

This utility represents an aggregation of all the attributes that characterize a design including performance metrics such as efficiency, maximum speed and power handling; market-driven metrics such as cost, durability and reparability; and consumer perceptions of the design state such as aesthetics and user-friendliness. These attributes have to be measurable in numbers and be reducible to a single value as a complete and final metric of a design candidate. In this dissertation, system dynamics performance are selected as the main attributes for design evaluation and very often are the most difficult ones since this demands a performance prediction by automatic modeling of dynamic system (or through physical experiments which, however, usually lead to substantial cost in labor and time).

As in Figure 1-2, once the design topologies are created, a system dynamics model for each design topology is generated by aggregating port based models of the components and the couplers between the components. It is very hard to decide the model complexity in terms of number of modes considered for each component at the conceptual design stage since most of the parameters are not available. As a strategy, the simplest model can be used for this design stage to predict the potential of a design. The

high fidelity models can be adopted for further evaluation for a small number of designs that passed the first-round screening.

1.2.4 Guiding Iterations

An exhaustive search for the best design is not possible due to the infinite size and complexity of the conceptual design search space. Fortunately, numerous optimization techniques have been developed which can provide a starting point in guiding the search for successful solutions. By using the utility function as the basis for comparison, techniques can be developed to efficiently find successful solutions without a complete search of the space. Stochastic searching, like genetic algorithms and simulated annealing can be applied to the problems that lack continuity and differentiability. Gradient-based optimization methods when applicable have an advantage with faster performance since the dynamics model is uniquely formulated with each topology, but have the drawback with only local optima reached within a multi-modal space. In this dissertation, a method of automatically preparing optimization is introduced, although it is not the focus of the major research efforts for this dissertation since searching algorithms already existing in the literature are used to guide the searching process.

In conceptual design, the utility function is not static due to new technology and processes, which implies that the global optimum is often in constant change. Therefore while optimization is a good starting point, one must realize that in true design nothing is fixed. The search space and the utility function change constantly, thus the search for the best design does not end accordingly. The real design process is marked by a series of "successful" design states. These successful designs, although prone to revision in the

future, emerge as a result of the design process converging on solutions that appear to satisfy the current utility function well enough to diminish the need for further search.

1.3 OVERVIEW

Thus far, three main theoretical challenges have been presented in developing an automated approach for optimal dynamic systems design: **creating topologies**, **evaluating topologies** and **guiding iterations**. In overcoming these challenges, A-ODDS integrates various facets, which can be viewed as three subsystems: 1) an evolvable agent - based algorithm, equipped with domain knowledge and decision preferences, that is responsible for **creating** candidate solutions, 2) a port-based method for automated modeling of physical design configurations and an automatic method for optimization formulation to allow for an adaptive approach to **evaluating** physical topologies, and 3) iterative algorithms (optimization) for **guiding** basic design concepts to successful design solutions.

The subsystem #2 combined with #3 (together task B in Figure 1-2) can be an independent system that, through automated modeling and optimization, provides a platform to test design concepts developed by human designers. When this test system is complete and stable, it can be connected to subsystem #1 (task A in Figure 1-2) to accept computer-generated design concepts for automated design that combines topology generation and evaluation. In the current implementation of A-ODDS research, task A and B are both researched theoretically and tested for effectiveness. The implementation of connecting task A and B is theoretically trivial but computationally highly demanding, and therefore is not within the scope of this dissertation work. Task C is not the focus of

this research since existing optimization algorithms are used, but certainly it is worthy of further work for better searching efficiency.

1.4 ORGANIZATION

In this dissertation a full description of the A-ODDS system is presented. The early chapters present the fundamental elements of the theory and the later chapters validate the various facets of the theory through test problems and experiments.

As seen in Figure 2, the next four chapters are devoted to the subsystems of A-ODDS. Each of these chapters presents the purpose, related research projects, and details of the operations of each subsystem.

Chapter 2 introduces the background of bond graph dynamic systems modeling. In Chapter 3, an evolvable agent based design method is proposed to generate design topologies. Design of low pass filters is used to evaluate the effectiveness of this design generation approach. Chapter 4 then presents the method of automated modeling to derive the mathematical model from a physical design, and the method to automatically prepare and invoke an optimization routine. A weighing machine design problem is used as the case study. Chapter 5 introduces concepts of model complexities, and the issues in model complexity are classified into two orthogonal planes according to the availability of detailed design parameters. Chapter 6 closes this dissertation by a summary of A-ODDS and a discussion of the theoretical claims it makes. Also, this chapter addresses conceptual design research challenges that emerge from the development of the A-ODDS methodology.

1.5 THESIS STATEMENT

An approach combining open-ended topology generation, automated modeling and optimization formulation, and iteratively guided searching is used for Automated Optimal Design of Dynamic Systems (A-ODDS).

Chapter 2 BOND GRAPH MODELING OF PHYSICAL SYSTEM

2.1 MODELING OF PHYSICAL SYSTEMS

A model of a given physical system is its functionally equivalent, simplified representation. Models can be of different forms, mathematical, graphical, schematic, prototype etc. Selection of a particular model for a given system depends on various factors. Prototyping a real system is expensive, sometimes hazardous and difficult to modify. On the other hand, a mathematical model can be simulated on a computer, is easily modified, and the effects of different parameters on system performance can be efficiently analyzed, although a prototype is always needed to finally prove the correctness of the model. The following is a list of basic definitions used in modeling theory. Refer to Figure 2-1.

- System

  A system is an entity that can be separated from the universe by means of physical or conceptual boundary. This comes from a thermodynamics perspective.

- Environment

  All that is external to the system is its environment. The environment can interact with the system. This also comes from a thermodynamics perspective.

- System variable

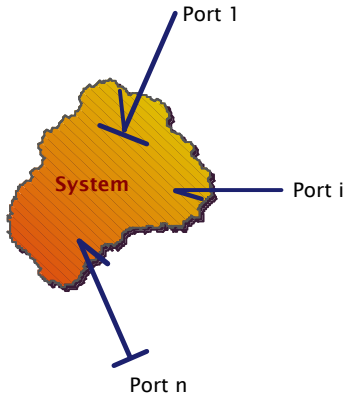


Figure 2-1: System Representation

Any characteristics of the system that changes with space and time is termed a system variable.

- Input variable

  This is a system variable which is manipulated by the environment without any influence from the system.

- Output variable

  Any system variables of interest that are measured and affect the environment.

- Model of a system

  A symbolic, graphical, physical, or mathematical representation of a system which describes the characteristics of the system variables.

- State variables

  These are the system variables which, different from system variables, define the "state" of the system of any time.

- State determined system

A system for which a set of state variables can be determined for all future times given the initial values of the set and values of all system inputs for future times.

2.2 BOND GRAPH MODELING TECHNIQUE

2.2.1 Introduction to Bond Graph Technique

Bond Graphs provide a structured approach to system modeling [Beaman and Paynter, 1995]. The complex physical system is split into simpler components, whose physical relations are more easily understood. These elements are interconnected to each other by bonds which represent power interactions. Power is expressed as a product of two variables associated with the bond. These are the generalized power variables, effort e (force, pressure, voltage, etc.) and flow f (velocity, flow, current. etc.). There are also generalized energy variables that represent the energy stored in the subsystem. The generalized energy and power variables are collectively called Bond Graph variables. Figure 2-2 shows a power bond and the relationship between these variables. Positive power direction is indicated by the half-arrow on the bond. Each element has a characteristic constitutive law, which relates its input and output.

2.2.2 Causality

Causality refers to the cause and effect. Assigning causality to a bond determines the input and output of effort and flow on the two elements connected by the bond. This is represented by a stroke at the head or tail of the bond known as a casual stroke. The convention for causal stroke is shown in Figure 2-3. When the causality stroke is near the

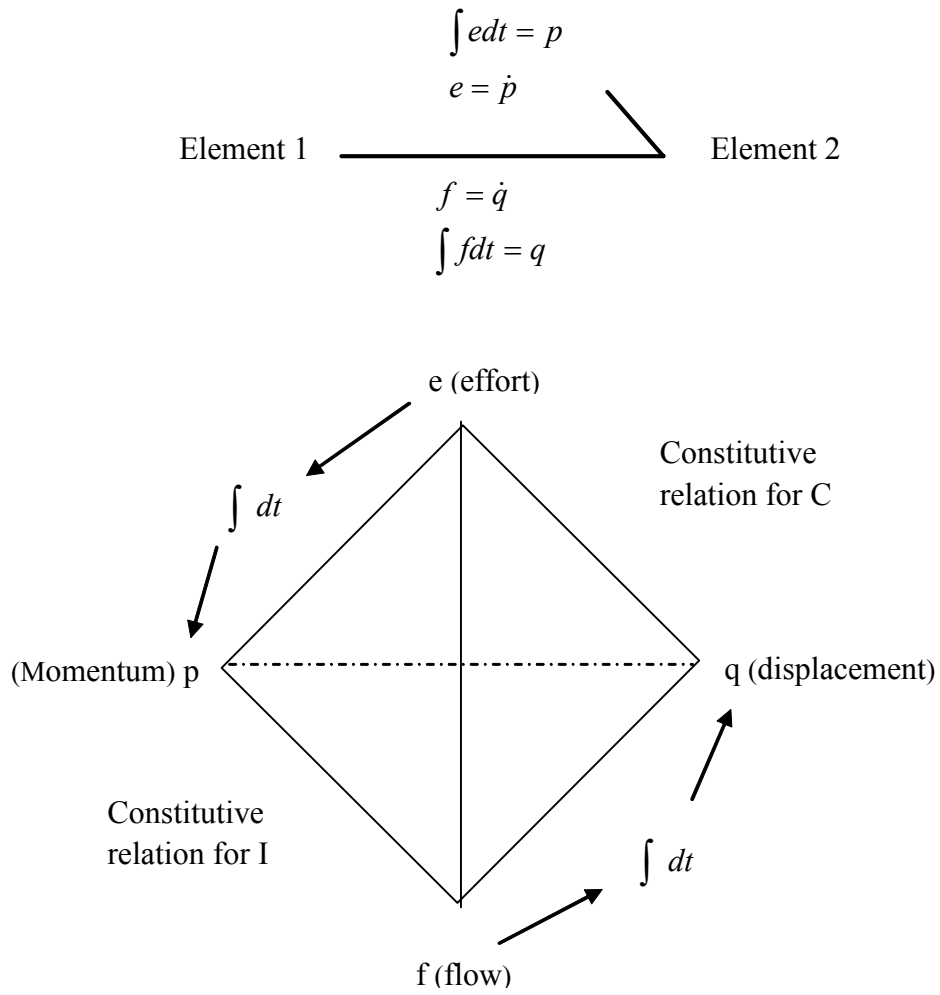


Figure 2-2: Power Bond and generalized Bond Graph variables

element (element B), it implies that the input to B is an effort and its output is flow, whereas a stroke away from an element (element A) implies that A gets a flow input and output effort. The only restriction on causality is that for any element either the flow or the effort can be specified. Both can not simultaneously be specified for an element. Along with this requirement, different Bond Graph elements have certain allowed and

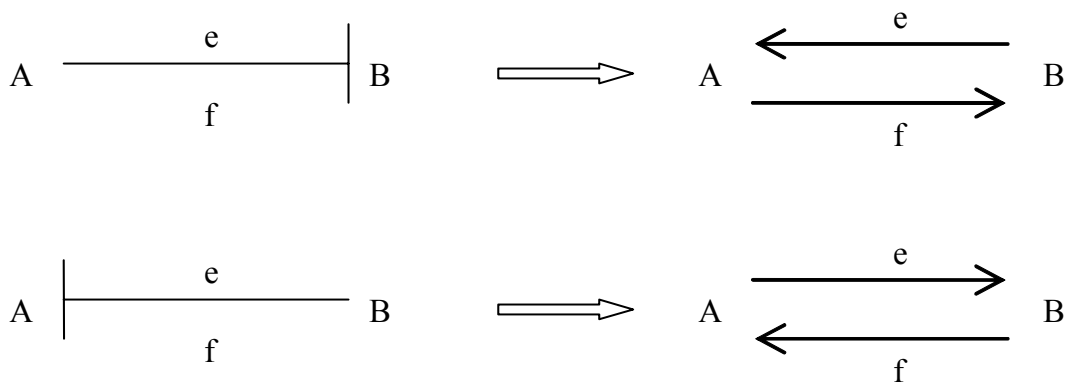


Figure 2-3: Causal stroke for a bond

preferred causalities. These are described for each of the element type in the following section. Assignment of causality in a Bond Graph directly leads to obtaining state equations.

2.2.3 Elements Types in Bond Graph

There are three basic types of elements.

- Energy Storage elements
- Transmission elements
- Dissipation elements

Different Bond Graph elements are now considered in detail.

- Capacitance, C

This is an Energy Storage element, that stores generalized potential energy. It is denoted by a 'C'. A C-element can be used to model mechanical compliance and

electric, hydraulic, thermal and chemical capacitors. The constructive law of this element relates generalized effort and generalized displacement. If one is an input,

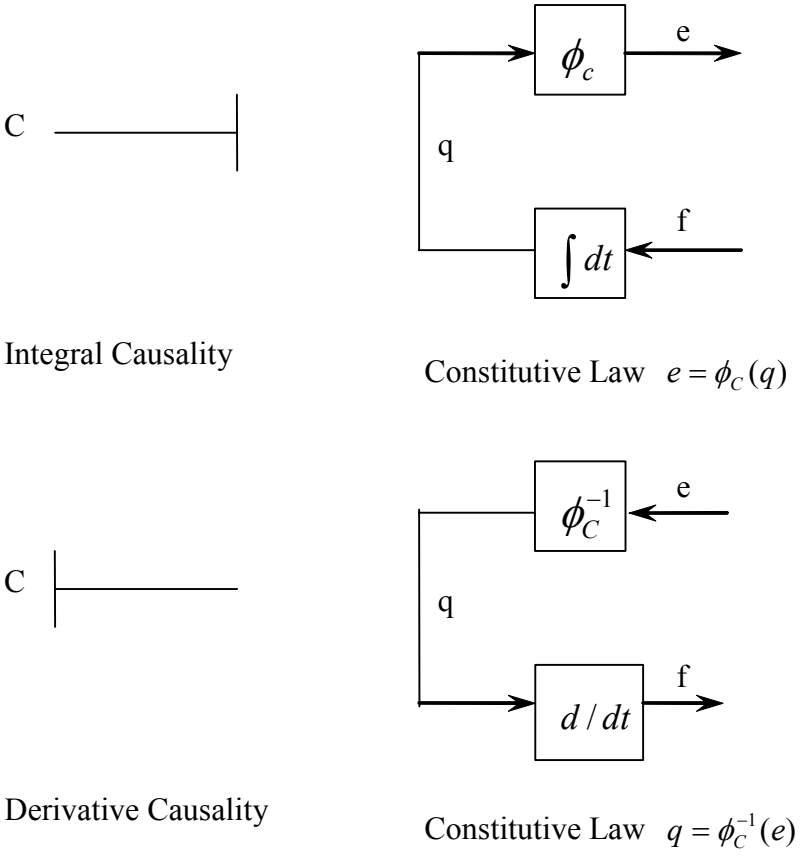


Figure 2-4: Generalized Capacitance

the other is the output from the element and vice versa. Depending on the input and output, a C-element can have integral or derivative causality. Integral causality is preferred, as it makes the element an independent energy storage element (determines the state of the system). The C-element with two types of causality and its constitutive function are shown in Figure 2-4.

- Inertance, I

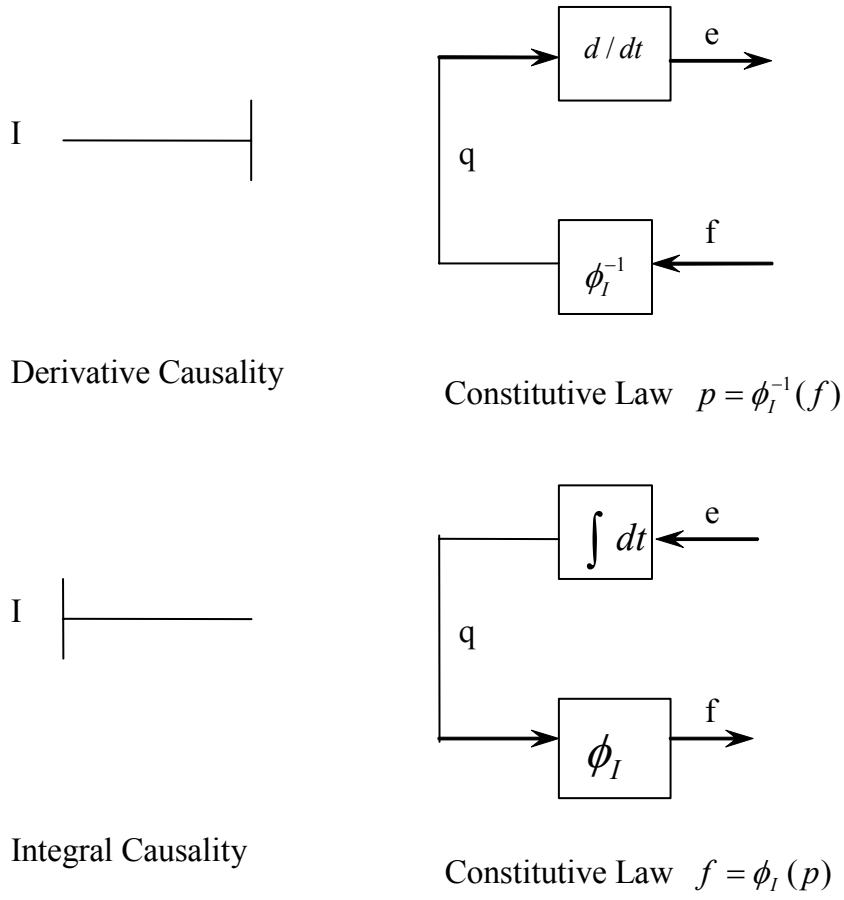


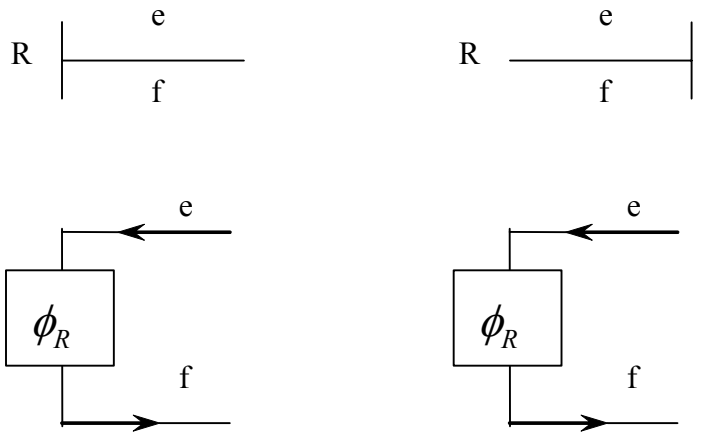
Figure 2-5: Generalized Inductance

This element stores kinetic energy. It is denoted by an 'I', and is the causal dual of the C-element. The I-element is used to model mass, inertia, and inductance effects in physical systems. The constitutive law for this element relates generalized momentum and generalized flow. If one is the input, the other must be the output and vice versa. The preferred causality for the I-element is again integral which makes it an independent energy storage element (determines the

state of the system). The I-element together with its causality and constitutive law is shown in Figure 2-5.

- Resistance, R

The resistance is a dissipative element. It is denoted by a 'R'. The R-element is used to model losses like friction, resistance etc. in a physical system. The energy dissipated by the R-element is given by P_d=e*f which is always greater than zero. The constitutive law of the R-element is a relation between generalized effort and generalized flow. Causality on the R-element can be imposed in either direction. If flow is the input and effort is the output, it is called resistance causality and if effort is the input and flow output, it is called conductance causality. Refer to Figure 2-6.



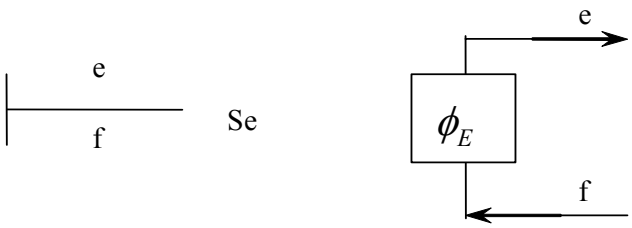
Conductance Causality $f = \phi_R(e)$

Resistance Causality $e = \phi_R(f)$

Figure 2-6: Generalized Resistance

- Effort Source, E

An effort source can be considered as a capacitance of infinite magnitude. It is denoted by a 'E'. The E-element is used for modeling a source of power flowing into the system. Since the E-element represents a source of effort, the causality assignment must be a stroke away from the bond as shown in Figure 2-7.



Constitutive law $e = \phi_E(f)$

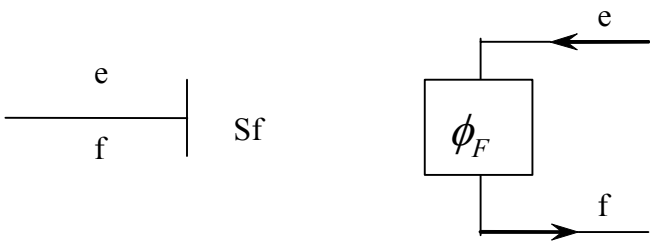
Figure 2-7: Source of effort

- Flow Source, F

The F-source can also be considered as an inertance of infinite magnitude. It is denoted by a 'F'. As for the E-element only one type of causality is allowed as shown in Figure 2-8, where in this case flow is always specified out of the F.

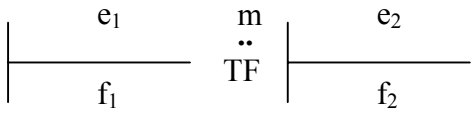
- Transformer, TF

The transmission element conserves power while transmitting it from one energy domain to another. It is denoted by a 'TF'. The TF element is used for modeling levers, gears, electric transformers, couplings etc. As shown in Figure 2-9, causality for a transformer can be assigned in only two ways.

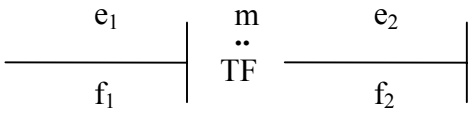


Constitutive law $f = \phi_F(e)$

Figure 2-8: Source of flow



Constitutive law $e1 = m * e2$, f2=m*f1

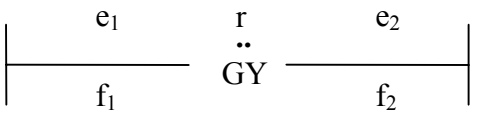


Constitutive law $f1 = f2 / m$, $e2 = e1 / m$

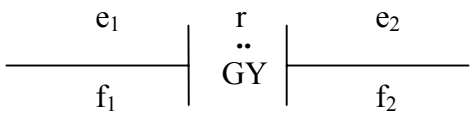
Figure 2-9: Transformer

- Gyrator, GY

The GY-element is a transmission element used for modeling isentropic effects in electromechanical couplings, gyroscopic forces. The constitutive laws and causality for GY-element is shown Figure 2-10.



Constitutive law $e1 = r * f2, \quad e2 = r * f1$

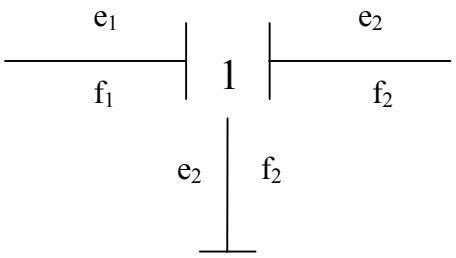


Constitutive law $f1 = e2 / r, \quad f2 = e1 / r$

Figure 2-10: Gyrator

- One Junction, 1

This element is basically a transmission element, but it is used for connecting other elements to form a system. It is denoted by a '1'. The 1-junction is used for joining processes like summation of forces and torques and series circuit connections. Functionally, it represents summation of efforts (given by Kirchoff's voltage law, $\sum Vi = 0$, written along a loop for electrical circuits), while maintaining flow constant on all elements connected to it (conservation of power). The causality for this element is thus restricted, because to make flow common to all bonds connected to it, flow has to be specified only once and on only one bond. All other bonds represent flow going out of the junction. This is shown inn Figure 2-11.



$$f1 = f2 = f3$$
$$e3 = e1 - e2$$

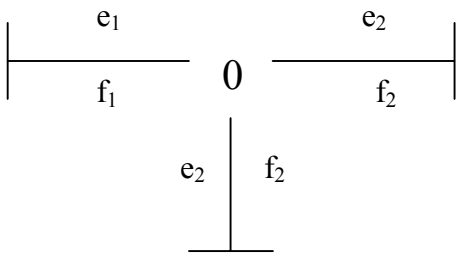
Figure 2-11: One Junction

- Zero Junction, 0

Like 1-junction, the 0-junction is also a transmission element used for connecting other elements to form a system. It is devoted by a '0'. It is used for joining processes like summation of velocities, parallel circuits and conversation of matter. It represents summation of flows (given by Kirchoff's voltage law, $\sum Vi = 0$, written along a loop for electrical circuits), while maintaining effort constant. The causality assignment for the 0-junction is as shown in Figure 2-12. (Only one effort input is allowed, all the remaining bonds specify effort out).

2.2.4 System Analysis

After constructing the Bond Graph for a given system, its time and frequency response can be obtained by systematically following a series of steps. These steps can be stated broadly as follows:



$$e1 = e2 = e3$$
$$f3 = f1 - f2$$

Figure 2-12: Zero Junction

- Causality assignment algorithm

  Using this algorithm, causality is assigned for the entire Bond Graph [Karnopp et al., 2000].

- State Equation generation

  Once the causality has been assigned, state equations can be formulated in a methodical manner [Karnopp et al., 2000].

- Analysis

  Finally the system response can be obtained from these state equations by processing them appropriately using various techniques of linearization, transfer functions, etc., [Karnopp et al., 2000].

A bond graph based model transformation tool (MTT) [Gawthrop and Smith, 1996] is introduced in Appendix A that can convert a bond graph to other representations such as transfer function, state-space equations, etc.

Chapter 3 AN APPROACH FOR SYSTEM DEVELOPMENT

3.1 INTRODUCTION

The evolutionary design method proposed in this research (the first loop in Figure 1.2) was inspired by the development process of a biological entity [Fontana and Buss, 1994]. DNA is not the only factor in determining the individual's phenotype (i.e. final form). Considering the amount of information contained in the genes and the level of complexity of the resulting individual, it is reasonable to conclude that there must be other factors involved in the cell's development. It is the authors' hypothesis that during development from a single cell, the creation of subsequent cells depends not only on the gene, but also non-deterministically on environmental conditions, and the prior history of how many cells and what types of cells have been produced. To model these unpredictable effects, *the gene is viewed as a probability vector for the generation of the phenotype*. Given this biologically inspired approach, the gene encodes not a solution to the problem, but instead a "generator" (in our case, a design agent).

Our design agents, composed of a probability based decision strategy and design grammars (production rules), are used to create a rich array of possible design topologies. A design agent provides design decisions throughout the search process. The decision strategy can be evolved using a genetic algorithm (GA) through which exploration and

exploitation of the design solution space are facilitated since the good strategies are selected into next generation and are further improved by crossover and mutation operations. Design grammars are predetermined for a specific application. Our goal is to prove the effectiveness of this approach and identify the key issues in advancing it toward becoming an effective and efficient open-ended topology generation method.

Genetic programming (GP) ([Koza, 1992], [Banzarf et al., 1998]) is usually used for system development, where the topology and parameter values are searched simultaneously through genetic operations. However, GP is inherently a stochastic approach that has low efficiency compared to deterministic searching methods [Papalambros, 2000]. The A-ODDS approach in this dissertation views the searching process as two separate stages, topology generation and parameter tuning, to allow flexible choice of the method for parameter tuning.

The bond graph is a modeling tool that provides a unified approach to the modeling and analysis of dynamic systems, especially hybrid multi-domain systems including mechanical, electrical, pneumatic, hydraulic, etc. [Karnopp et al., 1990]. It is the explicit representation of the system as a graph that makes the bond graph a good candidate for use in open-ended design searching. Genetic algorithms are an effective way to generate versatile design candidates evolutionally by genotype crossover and mutation. The design approach presented in this research, combines bond graphs for model representation and genetic algorithms for agent evolution as a means for automated dynamic system design.

3.2 BACKGROUND

3.2.1 Bond Graphs

Initial studies done for the purpose of dynamic system design use the basic bond graph elements: [C, I, R; 1, 0; Se, Sf]. Details of the bond graph notation and method were introduced in Chapter 2. For more description related to the bond graph, please see Karnopp et al. (1990) and Rosenberg (1975).

Using the bond graph (BG) formalism, a number of other approaches have explored the design automation of dynamic systems. Tay et al. (1998) used a genetic algorithm to suggest new dynamic system by topological remapping of system constituents (or bond graph elements). In this work, the algorithm starts from an existing BG design, and then applies a GA to generate new ones, which represents a new physical configuration according to some embedded or human preferred mapping mechanism. Seo et al. (2002) made further improvements on the dynamic system design automation. They used genetic programming (GP) and BG elements to dynamically generate BG designs from "embryos". The commonality of these two approaches is that bond graph designs are both represented genetically and evolved by genetic operators, while for the approach introduced in this dissertation the design agents are evolved instead of designs.

Rosenberg et al. [2001] identifies an interesting "catalog design" problem using bond graphs. This research continues along this direction by restricting the values of bond graph elements C, I, and R to a small set known a priori (e.g., a choice among a dozen R values). This becomes a practical design problem, but one with quite different features,

such as in this case the topology generation process can lead to a complete design without a need to decide values for elements of a design.

3.2.2 Genetic Algorithm and Genetic Programming

Genetic algorithms (GA) [Holland, 1992] search the solution space through simulated evolution. In general, the fittest individuals of any population tend to reproduce and survive to the next generation, thus improving successive generations. However, inferior individuals can, by chance, survive and also reproduce. GAs have been shown to solve linear and nonlinear problems by exploring all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population [Michalewicz, 1994].

In the genetic programming (GP), a hierarchical representation is manipulated by evolution, and a biologically-inspired encoding scheme is used to construct phenotypes. Many researchers have used genetic approaches to investigated automated design synthesis such as circuit design by Koza et al. [1997], robot design by Hornby et al. [2003] and sheet metal design by Patel and Campbell [2005]. In Koza's work, the genetic programming system begins with minimal knowledge of analog circuit design and creates circuits based on a circuit encoding technique. Various analog filter design problems have been solved using genetic programming, and an overview of these techniques, including eight analog circuit synthesis problems, is found in [Koza et al., 1997]. In genetic programming, the component values, design topology are evolved simultaneously.

3.2.3 Probability Vector

Population Based Incremental Learning (PBIL) was introduced by Baluja [1994] that combines aspects of genetic algorithms and competitive learning. PBIL's most significant difference from standard genetic algorithms is the removal of the population found in GA's while maintaining a probability vector. Other similar approaches include BOA (Bayesian Optimization Algorithm) [Pelikan et al., 1998], cGA (Compact Genetic Algorithm) [Harik et al., 1999] and UMDA (Univariate Marginal Distribution Algorithm) [Mühlenbein, 1998], etc. The A-ODDS approach in this dissertation is similar to those in terms of all using probabilistic techniques but different since this approach deals with open-ended system developments instead of static optimizations where the solution genotypes are of the same length.

More specifically, this approach encodes a design agent's probabilistic design strategy instead of a design. This design strategy, along with the design grammars, is used to develop designs probabilistically. The fitness of the designs generated from a certain design strategy suggests how this strategy (or this agent) performs, which is compiled as an evaluation routine for applying a GA to the design strategy. The proposed design cycle is shown in Figure 3-1. Given existing designs generated by the current design agents, we first check to see if the specifications are met. If so, the best design is selected and the design process ends; if the specifications are not met, the good design agents of the current epoch are selected and used to generate the next population of design agents.

Some extra introductions about the genetic algorithm based approaches can also be found in Appendix B.

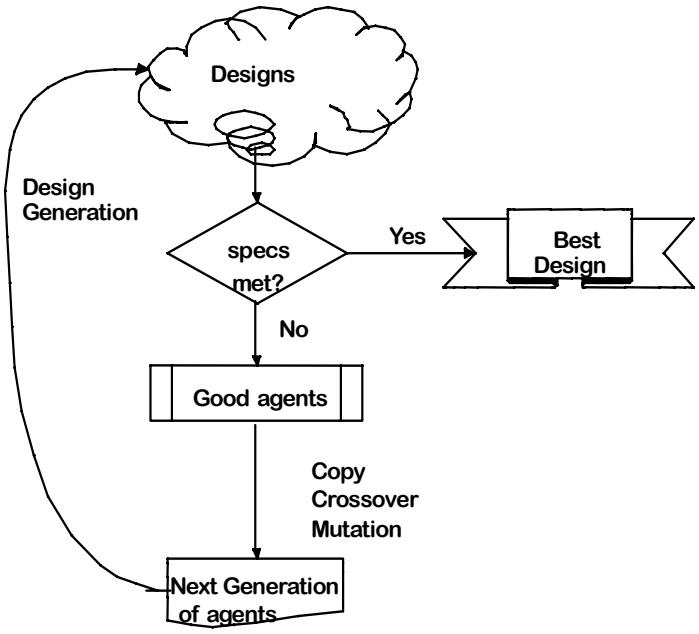


Figure 3-1: Proposed design cycle. On each generation the surviving design agents generate candidate designs (topologies). There is fitness evaluated for each design and for the design agents that are used to select the next generation of design agents.

3.3 REPRESENTATION IN GENERAL

3.3.1 Bond Graph Representation

For the purpose of multi-domain dynamic system design, the bond graph approach is adopted for model representation for the following reasons: First, bond graphs provide great convenience for modeling the wide range systems (for example, electrical, mechanical, hydraulic, pneumatic, and thermal) by using a common notation. Second, the graphical nature of bond graphs allows far easy generation by random combination of bond and node components, postponing the consideration of equation descriptions until needed for detailed analysis. Additionally, it is possible to span a

large search space of topologies with relatively few basic elements, or by combining the bond graph sub-modules of physical components. It is also easy to refine designs by operations such as adding size and/or complexity in any region (subsystem), deleting certain elements and/or replacing with others to meet performance requirements. Finally, the symbolic formulation of bond graph is ideally suited for parameter optimization at the later stage since gradient-based optimization is applicable and undoubtedly faster than stochastic searching methods.

In this initial study, elements `C`, `I`, and `R` are restricted to be linear one-ports with a fixed set of parameters. This element set together with the junction elements 0 and 1 are sufficient to allow us to achieve designs that have practical meaning in engineering terms, while we learn how the various factors of the search method influence the results. By arranging elements in series and parallel other values of "equivalent" elements can be achieved through evolution of the topology. In the future research, TF (Transformer), GY (Gyrator) and other bond graph elements can be added in the same manner to generate complicated multiple-energy domain system designs.

3.3.2 Probabilistic Design Strategy Representation:

The probabilistic design strategy is encoded using a list representation (Figure 3-2 (a)), which has a weight stored for each operand and operator. At each design stage, which operand or operator will be chosen is determined by the "spin of the roulette wheel". An element that has a larger weight has better chance of being chosen; hence these elements statistically contribute to the final designs more than some other elements. Some other items in the list representation include the expected complexity (number of

components) of a design, choice of an operand type, and choice of an evaluation method for a design or design strategy, etc. A design strategy (or a design agent) can produce good designs as well as bad designs, which is similar to human engineering design. Human designers need professional training to make good designs. Accordingly, the brain of a design agent, its design strategy, will be evolved by applying a GA to search for the design strategies that have the best chances of producing designs of high quality. The genotype has a fixed length, which conforms to the standard GA operation. From the pre-created design grammars, a gating mechanism (Figure 3-2 (b)) can be derived to aid in making decisions for a certain design context. The items in the list that are not plausible at a given design stage are gated (their probability is multiplied by 0.) For instance, to choose a component to connect to a motor shaft, the components without rotational ports are gated. When it comes to the time to decide an operator type, all the other unrelated items (such as operands) are not taken into consideration.

The gated probability (or weight) of a certain item in the strategy list is given by the multiplication of the value of this item and the gating factor of this item in a current design step. The probability to choose the item j in a gated design strategy is the normalized probability p_j of that item as shown in Equation 3.1.

$$p_j = \frac{P^j_{gated}}{\sum_{i=1}^{n} P^i_{gated}} \tag{3-1}$$

p (op1)	…	p (op*)	p (comp1)	…	p (comp*)	p (complexity)	…	p (type 1 operand)	…

(a): A probabilistic design strategy.

op1	…	op*	comp1	…	comp*	complexity	…	type 1 operand	…
1	0	1	0	0	0	0	0	0	0

(b): Gating probability vector.

Figure 3-2: Probability vectors. In (a) encodes the probabilities or weights of the design operators, operands, complexity, etc. The gated probability (or weight) of a certain item in the strategy list is given by the multiplication of the value of this item and the gating factor of this item shown in (b) in a current design step.

A design is 'grown' using the operators, operands and the operating locations iteratively determined by a design agent until a complexity limit is reached as is shown in Figure 3-3. The complexity limit of a design can be obtained through multiplying a complexity probability (an item of the design strategy) with a preset scale.

The agent fitness can be expressed as the fitness of the best design or the average of all designs it generates, or a hybrid of the two as expressed in the Equation 3.2.

$$f_{designer} = r * Best(F_{design}) + (1-r) * Avg(F_{design}) \qquad (3\text{-}2)$$

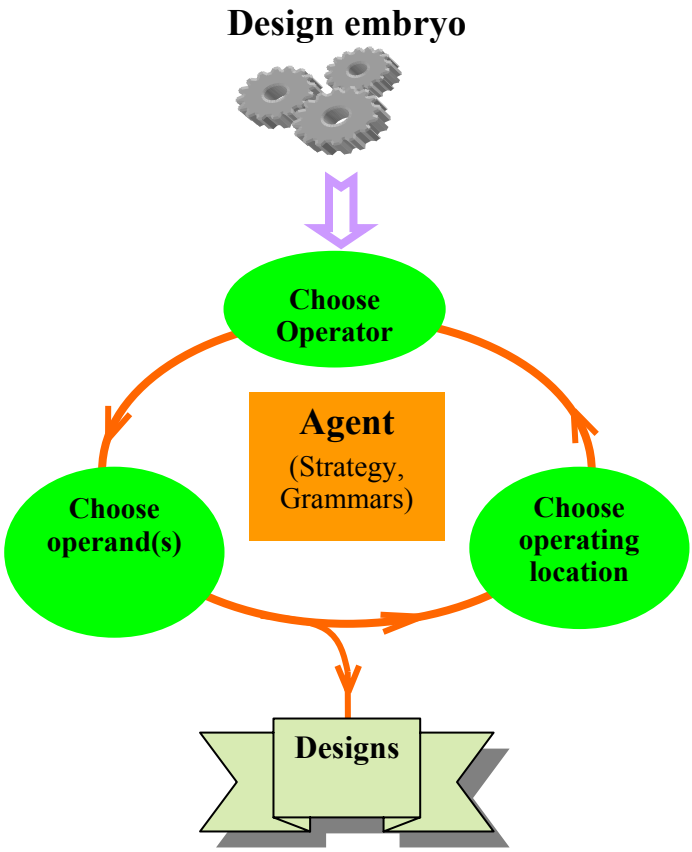
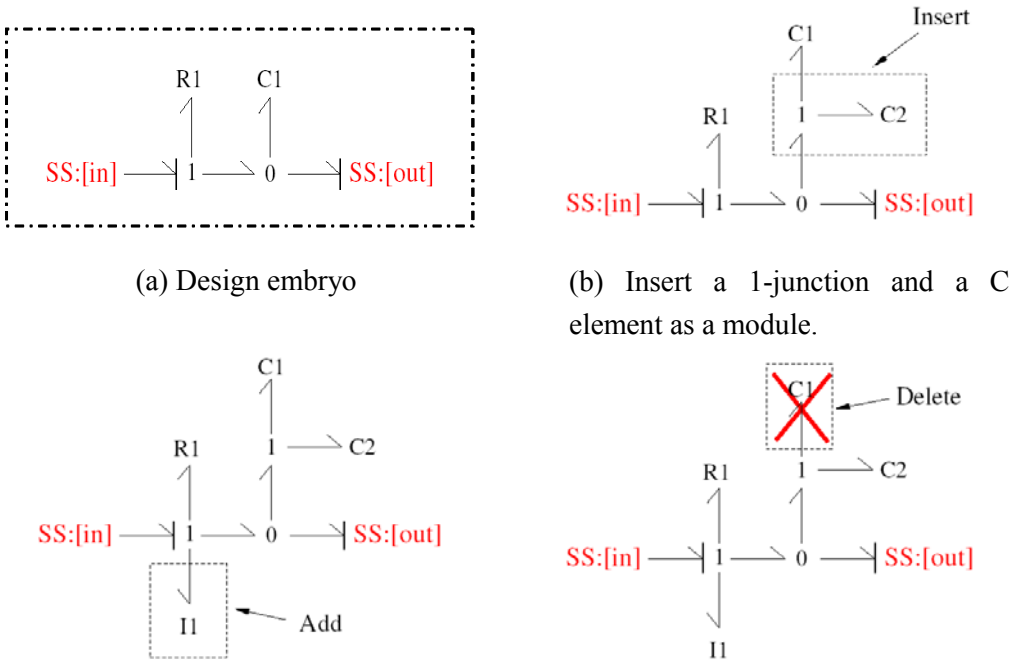


Figure 3-3: The process for design topology generation. For bond graph generation, the operators are Add, Insert, Delete, and Replace (Delete and Replace are mostly used in the application of modifying existing designs). the basic operands include elements 0 1, C, I, R, Tf, and Gy. The operating locations are the bonds or the junction elements (0 or 1).

3.4 GRAMMAR RULES FOR BOND GRAPH GENERATION

The design process can be initiated from an embryo bond graph to develop a new design or from a previous bond graph design to make modifications. Examples of applying basic operators are shown in Figure 3-4, where a design embryo (Figure 3-4 (a)) has an input and an output specified. The operations, 'Insert', 'Add', and 'Delete', are demonstrated in Figure 3-4 (b), (c) and (d) respectively in a developmental order.

Another operator ‘Replace’ is not shown, but similar as ‘Delete’, can be used to modify a design.



(a) Design embryo

(b) Insert a 1-junction and a C element as a module.

(c) Add an I element

(d) Delete a C element as a strategy to modify an existing design.

Figure 3-4: Applying basic operators to develop a design topology (bond graphs).

Bond graph design grammars are hierarchically represented as in Figure 3-5, which provides a gating mechanism and are used to guide the bond graph developmental process. To make an update to the design, firstly an operator is chosen probabilistically (represented as rectangles in Figure 3-5). The grammars on how to choose an operand for a specific operator are shown in Figure 3-5 (a). For an ‘Add’ operation, first a type of operand (noted in a double circle) needs to bespecified which could be a capacitance type (C) or a resistance (R). After a type is determined, an operand (noted in a single circle)

that has a value associated can to be chosen. For a 'Insert' operation, two operands are required with one of them as a junction element and the other as a C# or R# element that is attached to this junction element. The grammars on how to choose an operating location in a to-be-finished design are shown in Figure 3-5 (b). An 'Add' operator can only add an operand to a junction element (0 or 1) in a design; and an 'Insert' operator can only insert operands to a bond element by breaking this bond into two, which are connected to the inserted junction element. Combining the probabilistic design strategy and the grammar structure, the operand and operating location can be determined probabilistically.

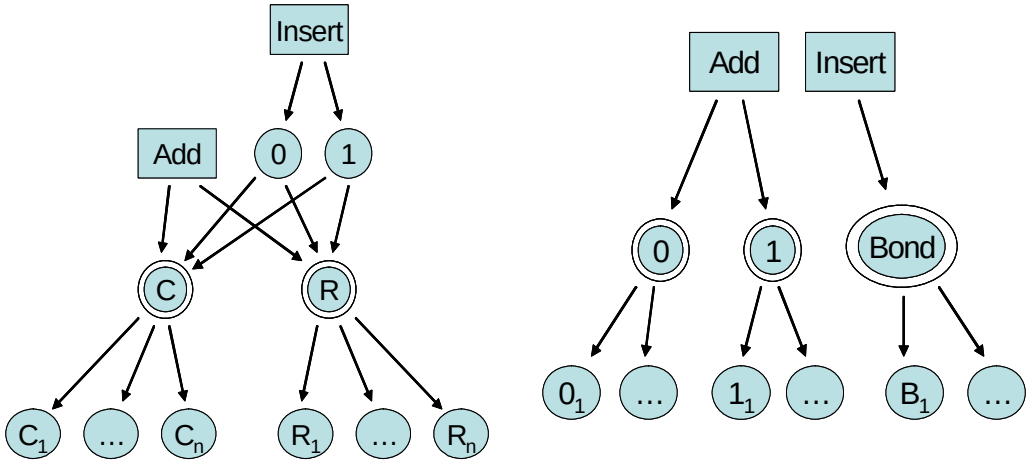


Figure 3-5: Hierarchical representation of the Bond graph design grammars.

3.5 CASE STUDY I: TOPOLOGY GENERATION OF A RC LOW PASS FILTER

A low pass filter design is used as a case study for the proposed approach. In reality it is common that a design engineer only has limited component resources, yet still

need to make designs by aggregating available components to meet certain requirements. The design effort in this case study aims to design a RC filter that has cutoff frequency as close as possible to a specified target, using only a number of types of resistors (each type has a unique resistance) and a number of types of capacitors (each type has unique capacitance). Topology generation is used to complete the design task without a parameter tuning stage.

3.5.1 Fitness Function of RC Filter Design

The fitness of a RC filter design is simply the absolute difference between the target cutoff frequency and actual cutoff frequency. The fitness of a design strategy is defined as in equation 3.2 with the parameter r set as 0, more specifically, the average fitness of the top one fourth designs it generated. Note that in this research, a smaller fitness means a better design or design strategy.

3.5.2 Probabilistic Design Strategy Representation

Three probabilistic design strategies are encoded in Figure 3-6 for the low pass RC filter design. The strategy 1 is a benchmark with same weight for every item. Strategy 2 and 3 are evolved through GA operations with the first population randomly generated. Following the bond graph design grammars hierarchically depicted in Figure 3-5, a design strategy provides a weight to each branch of a node in the grammar structure. The weights help determining the possibility in following a specific branch by computing the percentage weight out of the total weights of all the sibling branches, which is also captured in Equation 3.1 of the gating mechanism. There are different "zones" within the genotype to encode the likelihoods of which operator to apply (Add or Insert), which type

of operand to follow (C or R), and which concrete operand to choose (C#, R# or 0/1). Extensions can be made to include further information to help decide, for a chosen operator, an operating location in an uncompleted design. In the current implementation, equal weights are assigned to all the sibling branches shown in Figure 3-5 (b). Binary genotype representation is used for this example, where each cell in the table takes 4 bits to enable a value range between 0 and 15.

	C1	C2	C3	C4	C5	R1	R2	R3	R4	R5	C/R	0/1	Add/Insert
Strategy 1	7	7	7	7	7	7	7	7	7	7	7/7	7/7	7/7
Strategy 2	8	10	2	4	6	4	10	5	4	3	6/9	5/10	8/7
Strategy 3	11	5	0	10	1	15	12	4	7	2	7/8	0/15	6/9

Figure 3-6: Probabilistic design strategy encoding for the low pass RC filter design represented with bond graphs.

All the items of strategy 1 are assigned equal opportunity to participate a design practice when they are allowed grammatically. For example, when a type 'C' element (item C/R) is involved, all the branches (C#) starting from this node share the same weight. Strategy 2 and strategy 3 deviate from the equilibrium to give more preference to certain elements while putting aside others. In strategy 3, C3 is not deemed as an element that can provide a contributing capacitance value and hence assigned zero possibility to be chosen in a design. Another element, junction '0', also has a zero value, which means a parallel physical layout is not preferred at all.

3.5.3 Experimental Setup

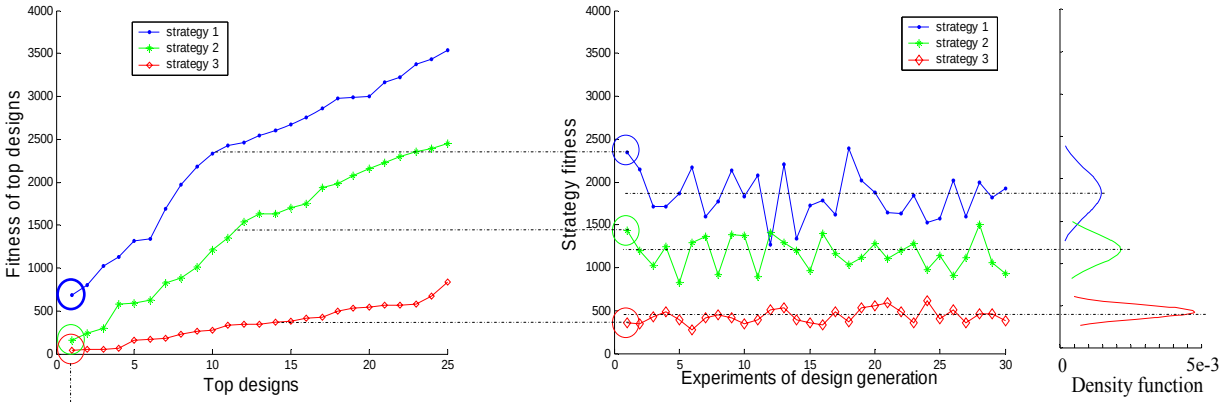
Our probabilistic strategy based design generation package was developed using Matlab®. A model transformation tool [Gawthrop and Bevan, 2003] was combined for

bond graph evaluation purpose. A genetic algorithm GAOT [Houck et al., 1995] was combined for the design strategy evolution.

3.5.4 Comparisons of Design Strategies

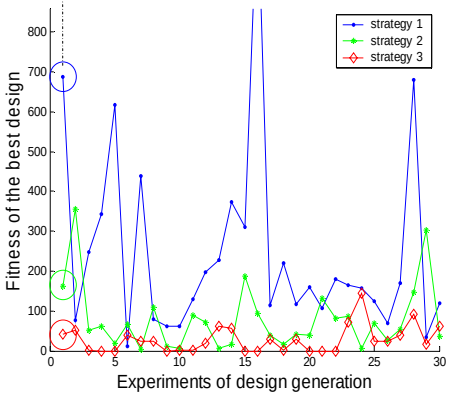
Comparisons are made between the three strategies shown in Figure 3-6 to design a RC low pass filter with a target cutoff frequency at 5000Hz (Figure 3-7). The average fitness of the top 25 out of 100 designs from each strategy is used as the indicator of this strategy's fitness (Figure 3-7 (a)). The best design of each strategy is the leftmost sample point marked out using a circle. The best design of strategy 3 has a fitness at 40.65 (the absolute difference between the real and target cutoff frequency), the best design of strategy 1 has an over 10 times larger fitness at 686.86 (the goal is to minimize fitness). The mean fitness of the 25 designs of each strategy is indicated by a dotted line, by which the average fitness of the strategy 3 is over 6 times less than that of the strategy 1. Strategy 3 shows better performance than the other two.

However, one comparison does not provide enough information to conclude that strategy 3 is consistently better. To verify the consistency of a strategy's performance, each strategy is executed 29 more times. Every time, the average fitness of each strategy is added into Figure 3-7 (b). The experimental results show that both the fitness mean and standard deviation of strategy 3 are over 4 times less than strategy 1 as shown in the distribution analysis at the right of Figure 3-7 (b). This proves that, relatively speaking, a design strategy creates a predictable set of solutions. Strategy 3 always has the best fitness while strategy 1 is always the producer of the worst.



(a) Comparison between strategies by the quality of top designs.

(b) Consistency study of the strategies' performance through 30 experiments of each design strategy.



(c) The best design from each of the 30 executions of each design strategy.

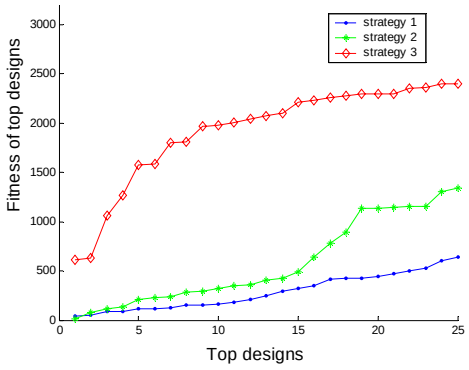
Best Design	Strategy 1	Strategy 2	Strategy 3
# of Designs 1% inaccuracy	2	11	24
# of Designs 0.1% inaccuracy	0	2	12
# of Designs 0.01% inaccuracy	0	0	12

(d) Comparison in terms of number of good designs generated by the three strategies.

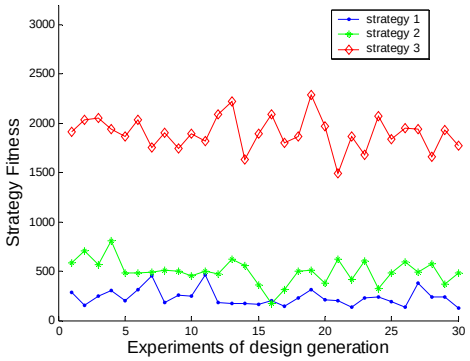
Figure 3-7: Comparisons between three design strategies to design a RC low pass filter with a cutoff frequency at 5000Hz. Note that in this research fitness reaches the best when it equals zero.

In addition to the average fitness of the top designs, it is also important to compare design strategies in terms of how easy it is to generate designs that have the best fitness (ideally equal to zero). The experimental data on best fitness from last 30 comparison executions are shown in Figure 3-7 (c), which indicates that strategy 3 generates the best designs most often, although occasionally it is outperformed by other strategies. Strategy 1 never reaches the zero line and almost never beats strategy 2 and 3. A detailed comparison as in Figure 3-7 (d)) shows the effectiveness of different design strategies. The number of designs for each strategy that reach a specific accuracy requirement is counted. Strategy 3 has 12 designs generated that have less than 0.01% inaccuracy, while strategy 1 and 2 do not have any, which clearly proves that strategy 3 is the most effective in generating successful designs.

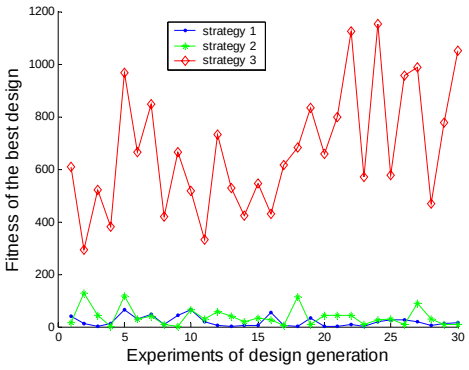
Another set of comparisons are made in a similar fashion between the same three design strategies as in Figure 3-6. The difference is that this time the designs' target cutoff frequency is at 300 Hz instead of 5000 Hz. However, with this target change, strategy 3 creates low performance solutions. The designs it generates have relatively intermediate designs as depicted in Figure 3-8 (a). This low quality is further confirmed by the consistency test of design strategies through 30 times of executions of each strategy as in Figure 3-8 (b), where the diamond red line (strategy 3) is significantly above the other two. Further, the best design that the strategy 3 has reached is also not as good as the ones from the other two strategies (Figure 3-8 (c)). The fact that a design strategy has different performances for different design targets suggests that evolutionary design of design strategies is beneficial to a design agent to utilize its design grammars more efficiently.



(a)



(b)



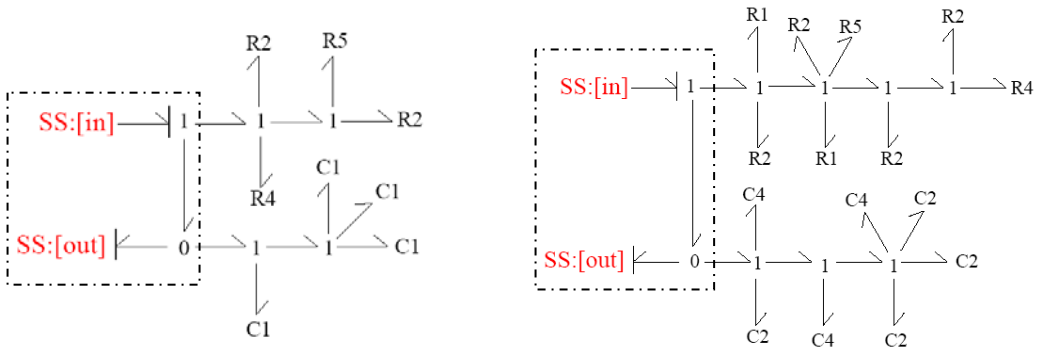
(c)

Figure 3-8: Performance comparisons between the same three design agents but with the target cutoff frequency of a low pass RC filter at 300 HZ instead of 5000 HZ.

3.5.5 Design Results

The probabilistic strategy based evolutionary process for RC filter design has the target cutoff frequency at 5000 Hz. The best design was created by a design strategy at the 39th generation of design strategies. This design strategy was actually the one introduced previously (strategy 3 in Figure 3-6). The two best designs (both with less than 0.0001% error) are showed in Figure 3-9 (a) and (b) respectively, which are among

the 12 designs in the bottom-right cell of Figure 3-7 (d). Each C# and R# has a property as specified in Figure 3-9 (d). Note that these two designs have only junction '1' since the design strategy 3, by encoding, does not view junction '0' as a useful element. This



(a) (b)

(c)

Comp	C1	C2	C3	C4	C5
Value (μF)	0.01	0.05	0.1	0.5	1

Comp	R1	R2	R3	R4	R5
Value (kΩ)	0.1	0.5	1	5	10

(d)

Figure 3-9: The best designs generated by design strategy 3 (a) and (b) and strategy 2 (c). The property values for all the components used for the RC filter design are shown in the table.

physically means that strategy 3 "believes" the target value can be achieved by connecting physical element (R# and C#) serially. Strategy 2 also generates a good design (Figure 3-9 (c)) with a 0.012% inaccuracy and its design has a combination of serial (0 junction) and parallel (1 junction) connections. In this design practice, the design complexity is allowed to take any integer value within a range to test the general performance of a design strategy. An alternative is to set complexity as a probability item in the strategy representation. Then the expected complexity limit of a design can be obtained through multiplying this probability with a preset scale. Also note that it does not hold that a design strategy can deterministically generate a same design next time since what is encoded in an agent are the probabilities for different decisions at each design step. The LPF design of this section exemplifies the "catalog design" problems (element values are limited to a small set known a priori), the design task is solely completed through topology generation without parameter tuning.

3.6 CASE STUDY II: DESIGN WITH PARAMETER TUNING

Generally speaking, there are two approaches for design space searching to meet design specifications, stochastic searching (such as genetic programming [Koza, 1992, 1994a, 1999], and gradient based searching [Papalambros, 2000]. Genetic programming stochastically generates topologies and tunes parameters at the same time [Seo et al., 2002; Koza, 1992]. The A-ODDS approach introduced in this dissertation is similar to genetic programming for topology generation in using stochastic searching, while different on the parameter tuning since our approach deals parameter optimization as a

separate stage that is open to using gradient based optimization (when model available) to reduce the computation time substantially.

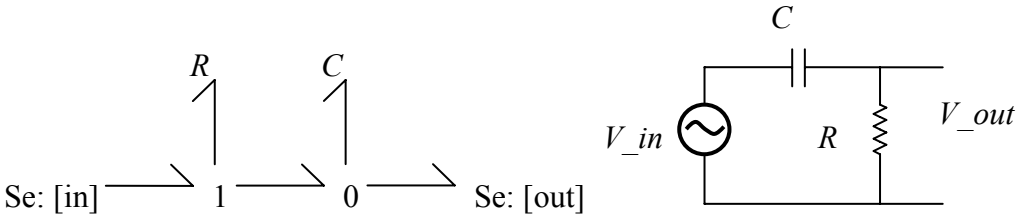
This case study also relates to the design of a low pass filter, but the design goal is changed from finding the closest cutoff frequency to finding the best overall performance and an extra element, inductor (I), is to be used. The optimization toolbox from Matlab® is used to optimize the parameter values of the design topologies. These topologies are developed using the same approach as in the previous case study, but this time any value within a range, instead of only a limited of number of values, is allowed for each type of component.

An ideal LPF filter has no effects to the portion of the input signal below the cut-off frequency, but completely damps the signal at all other frequencies, like the green step function shown in Figure 3-12. A LPF design is evaluated by the squared sum of the difference at a certain number (n) of distributed sample points as shown at Equation 3.3 with y_i as the magnitude at the i^{th} sample spot from the ideal LPF response while $f(x_i)$ from our design. The fitness of an agent is defined as the fitness of the best design that the agent generated, where the fitness of a design is based on minimizing the fitness function that follows.

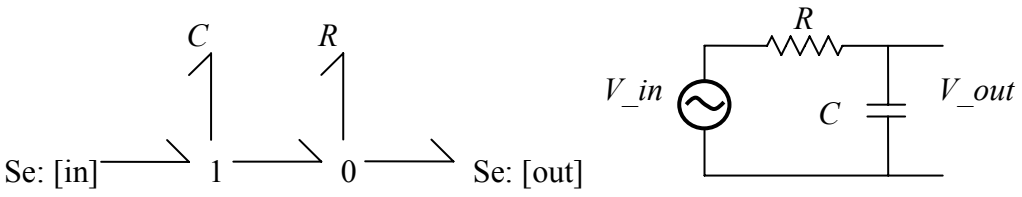
$$Minimize: \qquad difference = \sum_{i=1}^{n} [y_i - f(x_i)]^2 \tag{3-3}$$

In Figure 3-10, (a) is one of the design candidates for low pass filter design with both the bond graph and the mapping circuit shown, but its topology determines that there is no parametric solution for R and C to be able to meet the design requirements as

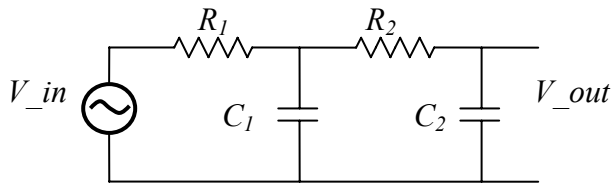
described above. Its typical magnitude frequency response is shown as the dashed curve in Figure 3-11. The green lines are the target design behavior (7Hz cutoff frequency) from an ideal low pass filter. In Figure 3-10 (b) a slight change is made by switching the position of the R and C elements. Known the target time constant $\tau = RC = \frac{1}{7}$, the values of R and C can be adjusted to obtain the cutoff frequency 7Hz. However, the best performance this topology can achieve as shown in the blue solid curve of Figure 3-11 is



(a) A design not fit for a low pass filter.



(b) A design for a low pass filter.



(c) Two circuits in (b) stacked for a higher order low pass filter.

Figure 3-10: Low pass filter design candidates.

limited by its inherent low-order nature so that the design behavior is always not good enough compared to the target behavior. Figure 3-10 (c) shows another topology that has two circuits shown in (b) stacked together to increase the system order. Although the performance curve (red '+' in Figure 3-11) becomes steeper, the inputs with frequencies smaller than the cutoff frequency are still damped significantly and the inputs with frequencies greater than the cutoff frequency are still not damped enough, which is not desirable. This indicates that other higher-order topology candidates need to be explored and topology design is critical to generate successful final designs.

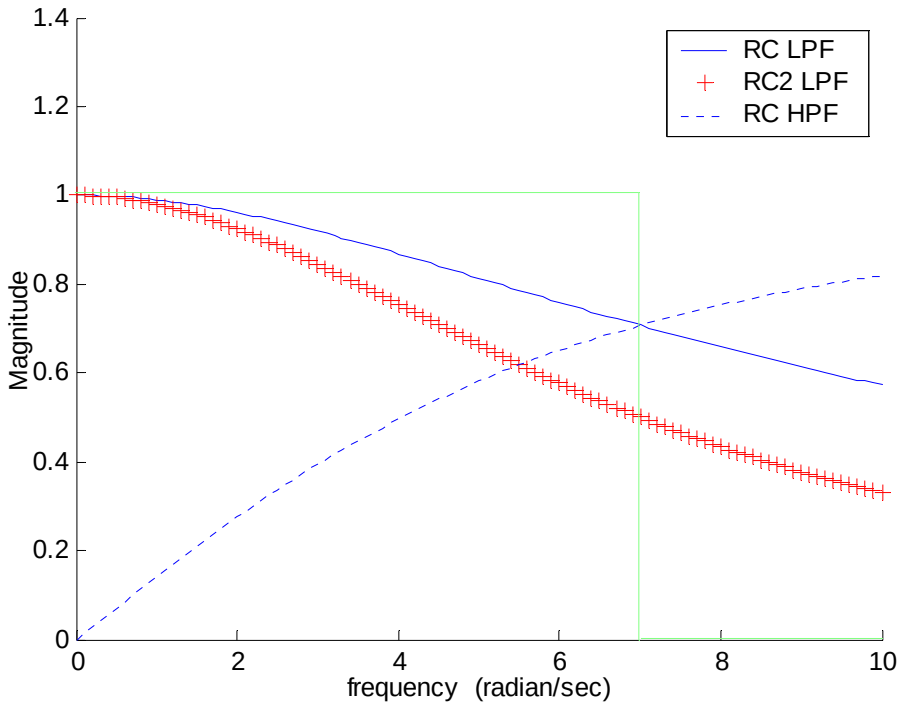


Figure 3-11: Magnitude frequency response

Shown in Figure 3-12 are the magnitude frequency responses of the evolved best low pass filters. The curve with red '+' has the best performance. The tuned best bond graph is shown in Figure 3-13 (a) with all parameter values marked in SI units. Figure 3-13 (b) shows the circuit that maps this bond graph. The bond graph topology is generated following the agent based design approach as described in Figure 3-3. For each generated topology, parameter tuning only takes a few seconds to find a local optimum and reasonable confidence is gained by starting searching at distributed locations in the solution space for each design topology. From our experiments, using genetic algorithm for parameter tuning does not give satisfactory results since it takes significantly longer time to achieve a design with the same performance.

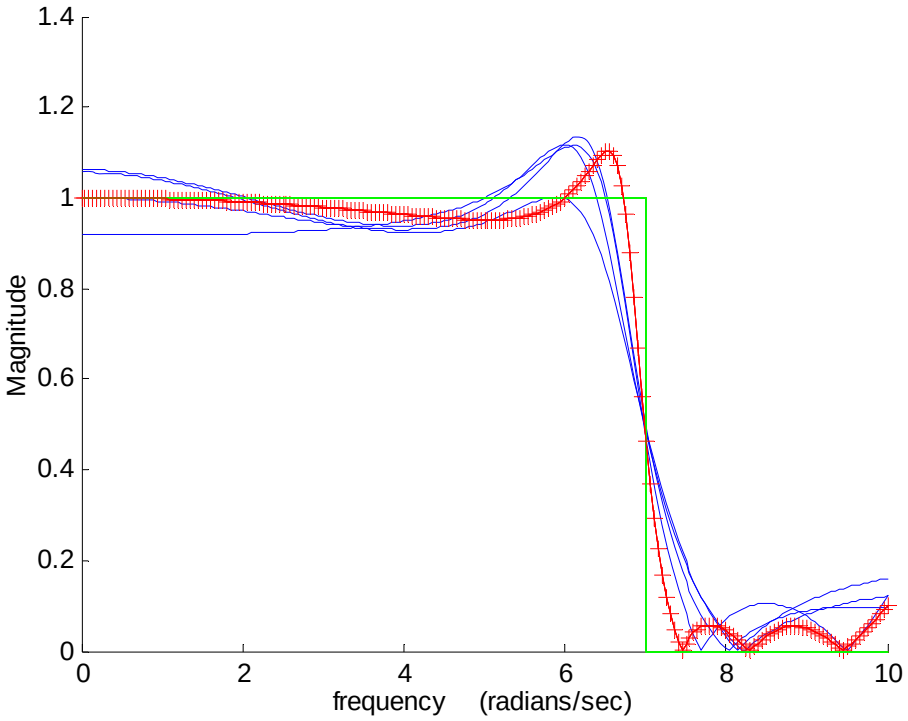
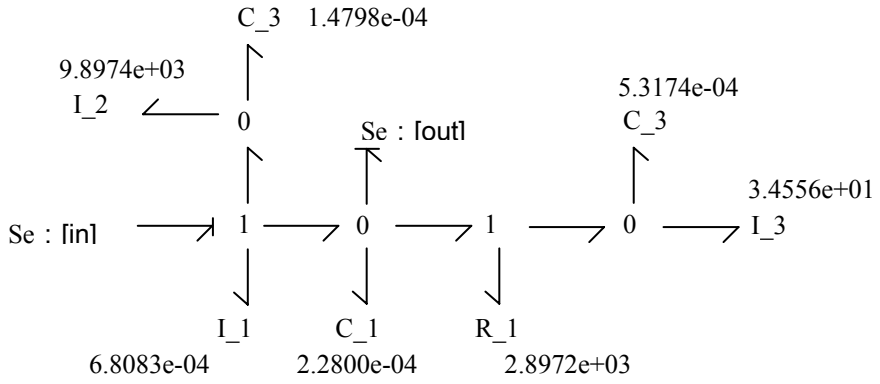
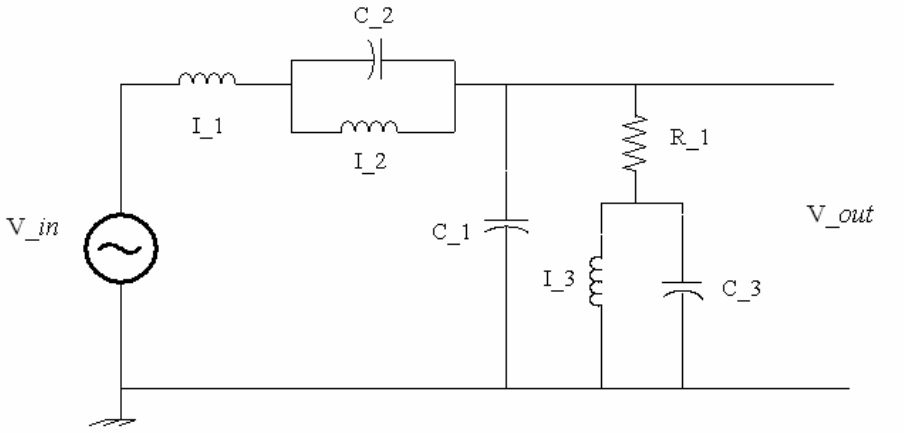


Figure 3-12: Magnitude frequency response



(a): Bond graph of the best performance



(b): The Circuit that maps the bond graph in (a)

Figure 3-13: Low pass RLC filter design.

3.7 DISCUSSION & CONCLUSION

In Genetic Programming, crossover operator selects two parent designs independently, based on fitness, and swaps randomly chosen portions of genetic material (subtrees). In this process, crossover could introduce non-existing component values and unexpected component types into a tree branch. Although this can be solved by a strong

typing mechanism, the algorithm usually becomes very complicated for implementation. Mutation operator creates a new design by mutating a randomly chosen part of an existing design. Mutation usually only accounts for a small portion of the total genetic operations (below 10%) in generating a new population. When the mutation rate increases and reaches 100% (no crossover any more), it becomes the approach introduced in this research that use evolvable design agents for complete regeneration at each epoch.

This research proposes an evolutionary design method that combines a probability based decision strategy and design grammars (production rules) into a "design agent" for system development. The decision strategy can be evolved by applying a genetic algorithm (GA) to facilitate the exploration of a multi-domain design space in a topologically open-ended manner, and still efficiently find appropriate design configurations. This method is applied to make dynamic system designs using bond graphs. Experimental results in designing RC low pass filters show that design strategies demonstrate steady performance in terms of the overall fitness of its top designs. A good design strategy/agent has a better chance of producing superior designs.

Unlike genetic programming where topology generation and parameter tuning happens at the same time, in A-ODDS research they are separated as two consecutive stages, which allows flexible choice of the optimization method for parameter tuning. This tuning method can be, generically speaking, stochastic or deterministic or a hybrid of both for better performance in terms of design quality and speed. Experimental results by RCL low pass filter design show that with gradient based optimization this two-stage approach can be tremendously faster than regular genetic approaches on reaching designs of the same quality.

With regard to the empirical results, it should be noted that it is incorrect to say the approach introduced in this research will always outperform a traditional GA or GP that directly encodes designs. In complex nonlinear system development, GP has relatively good performance where the design space becomes highly rugged, non-continuous and gradient based nonlinear optimization can not work well. However, this approach is much simpler than GP in terms of understanding and implementation for open-ended system development since there will be no genetic tree operations, no strict type matching and less pre-mature convergence (GA applied on design agent level instead of designs).

A parallel research in the dynamic system design is to create a repository of electro-mechanical components (such as gear, shaft, motor, spring, bar, etc.) that stores the knowledge (grammar rules) of how interactions between components happen and how to model the components and the interconnections using bond graphs within various coupling contexts (next chapter). This repository will provide a broader application for the proposed design method. Additionally, the design strategy list representation remains extensible to meet other design needs by adding decision variables. Further, some other mechanisms (such as the incremental learning) can be used to replace GA to update /evolve a design agent/strategy in the future work of this research.

Chapter 4 AUTOMATED MODELING FOR DESIGN EVALUATION

4.1 INTRODUCTION

Conceptual design of dynamic systems requires the designer to assemble a multitude of parts selected from a large discrete set of components to perform a function. The designer must satisfy a variety of constraints (limits, inequalities, etc.) on the overall design as well as for the different components, while reconciling conflicting cost functions. During the initial conceptual design phase, detailed values about each component are not known but rather a simple connectivity of components is described. Typically the configuration is divided into subsystems and different engineers (teams) are assigned to develop each subsystem of the device. Each team receives newly specified requirements that allow the finished subsystems to come together into a complete product. Often, such "over-the-wall" design practice can lead to final designs that are cumbersome or lack robustness in meeting all the originally specified design requirements. It is only through various redesigns that the design team(s) is (are) able to see the subtleties in how various subsystems interact and how the individual components can be optimized to arrive at a more elegant solution to the original design problem.

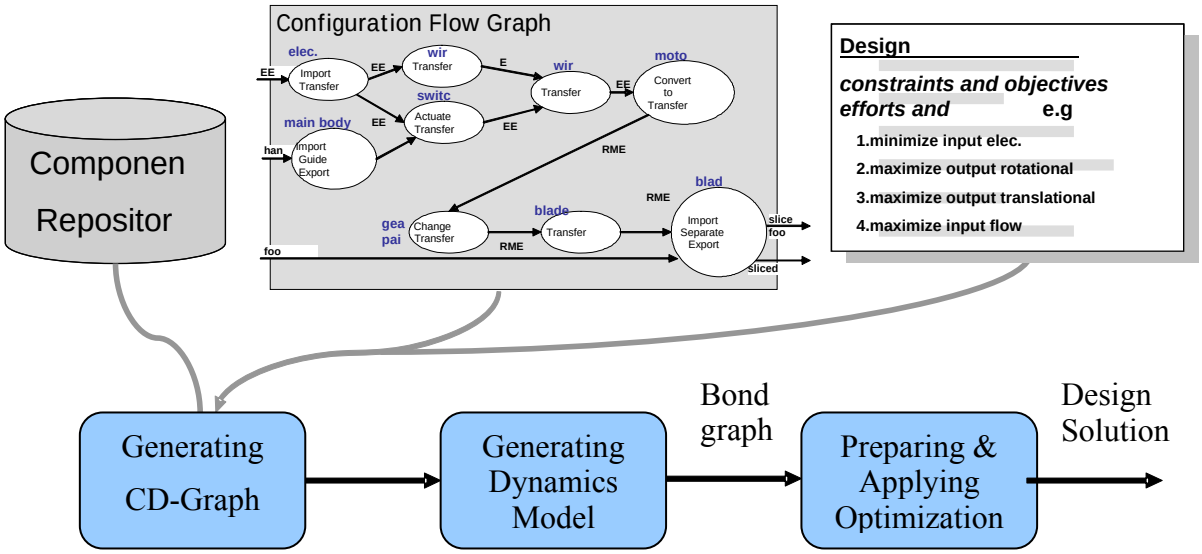


Figure 4-1: Graph depicting the assistive design tool introduced in this Chapter. Given the design specifications (white box), the designer conceptualizes a design (CD-Graph) based on experiences or via a configuration flow graph (CFG) generated computationally [Kurtoglu et. al, 2005] using the components in a repository. Bond graph based generic models are introduced that facilitate the automated modeling of design configurations. An approach for automatic design genotype preparation provides convenience to the subsequent optimization task that outputs detailed design solutions using genetic algorithm (GA).

As a solution to this typical scenario, a computer aided design tool is envisioned as shown in Figure 4-1. This tool is a subset of the overall approach depicted in Figure 1.2 (not including the first topology generation loop that was explained in Chapter 3 in detail). Instead of computer generated design topologies, given the design specifications, engineering designers would draft an entire configuration or topology of their concept perhaps from a function structure [Kurtoglu et al., 2005] as in the upper-left block of Figure 4-1 by simply "dragging and dropping" elements from a component repository. The details of the coupling between any two components are captured by a multi-domain design representation referred as Conceptual Dynamics Graph (CD-Graph). There are

various ways for two components to be coupled, and the coupling format could be decided intelligently by the automatic detection of connection compatibility between components, which can be further compensated with designers' instructions if necessary. This topology generation process can also be automated using the design approach proposed in last chapter. The focus of this chapter is how to evaluate a design configuration automatically.

The next block, "Generating Dynamics Model", aims at obtaining the system dynamics model based on a conceptual design represented by a CD-Graph. Once the system model becomes available, the transformation from the system model (e.g. state space representation) to another desired representation (e.g. transfer function) can be automated. Since design goals are clearly defined prior to any computational design effort, so are the strategies that are used to evaluate the design's performance. The design performance, as outputs, is determined by the inputs and the system transfer function. Hence the conclusion comes that the design evaluation function (or the objective function) can be obtained automatically from a CD-Graph. In this research, automated evaluation is demonstrated through automated bond graph modeling and model transformation according to the goals defined by designers.

The last step, given the evaluation function, is to automatically invoke an optimization process to determine the choices for the design variables in each of the components pulled from the repository into the configuration. Design variables can be decision variables (e.g. length of a bar component) or a dependant variables (e.g. stiffness of a spring component). This section discusses a systematic method to automatically prepare a design problem for the application of optimization using genetic algorithm.

This preparation sets up the genotype representation for a design by encoding design variables, while taking into consideration of the design constraints and physical constitutive laws that were pre-specified in the component repository.

4.2 RELATED WORK

A requirement in the design of dynamic systems is the availability of dynamics models of individual components and an effective mechanism to assemble them into a consistent model. In order to allow for automated modeling of electromechanical systems, a detailed library of components must be accessible for reference to build aggregate dynamics models. Given the large number of OEM (original equipment manufacturer) parts and custom components that can be used in a given configuration, it is a difficult task to construct a library that is useful to designers. Fortunately, the construction of such a component repository is well under way ([Szykman et al., 1999]; [NIST, 2000]). Researchers at The University of Texas at Austin, University of Missouri-Rolla, and other institutions, have been building the details of this repository by dissecting and recording each individual component in a given artifact [Bohm et al., Currently within the repository, a set of characteristics (Figure 4-2) exist for each component including information about its form, function, dynamic behavior, etc.

The bond graph (BG) formulation is used for the dynamics model development because it facilitates the integration of component/subsystem models, provides the user with physical insight, and allows easy manipulation of models. A "word bond graph" [Karnopp et al., 1990] is a less detailed, higher-level representation, where major subsystems are represented by words. Figure 4-3 shows a word bond graph example of

a drive train. The representation is compact and provides only key information for design. Once a bond graph is created for each subsystem, the system bond graph model of the drive chain can be generated using the interconnections (power bonds). However, without a detailed design description, engineering assumptions (such as the coupling type between the shafts and the belt drive) need to be used, which becomes especially true when dealing with problems with multiple domains (e.g., 2D motion with 3 domains of x, y, θ_z instead of only 1 rotation domain shown in Figure 4-3). In modeling practices,

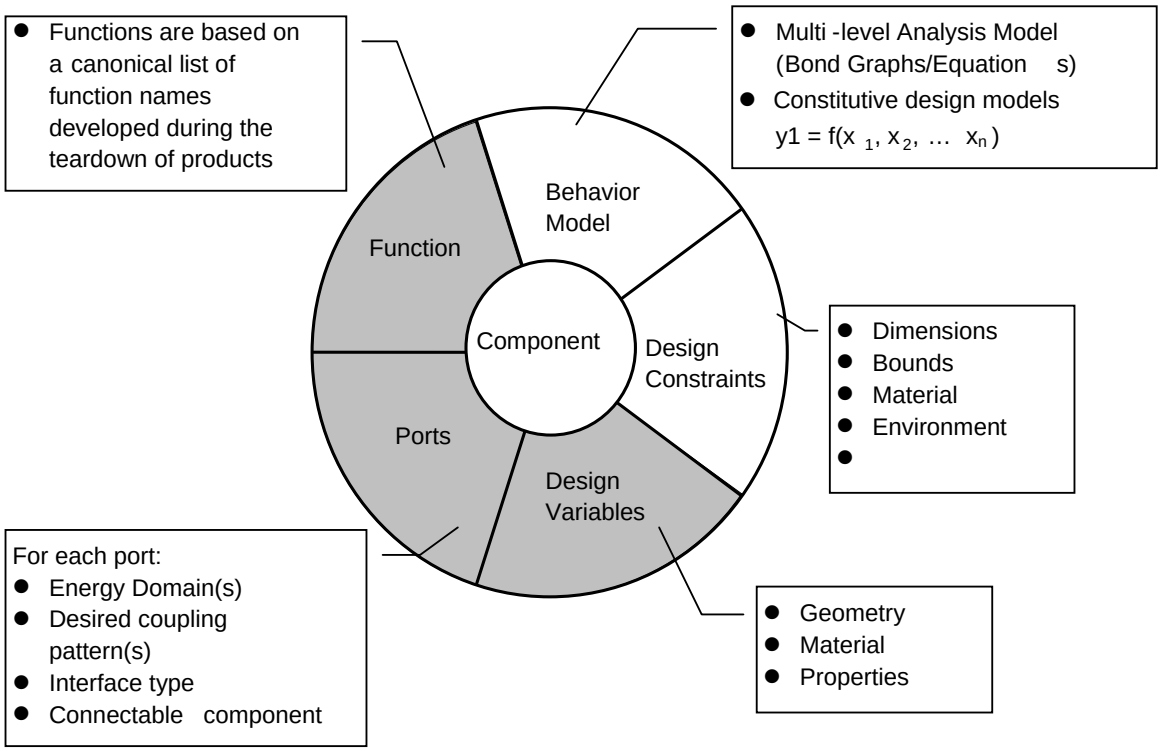


Figure 4-2: Component Repository. Within each component in the repository, there are five pieces of information stored. In this research, information about behavior model and design constraints was added for system modeling and tuning of design parameters.

word bond graphs are more used as an assistive strategy to modularize the system for modeling instead of a systematic approach for automated modeling of design configurations.

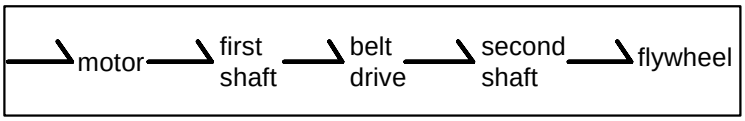


Figure 4-3: The word bond graph of a drive drain.

A number of other approaches explored the design automation of dynamic systems. Using a bond graph (BG) formalism, Tay et al. (1998) used a bond graph method with a genetic algorithm (GA) to construct dynamic system model. Seo et al. (2003) made further improvements that used genetic programming (GP) and BG elements to dynamically generate BG designs from embryos. The results from both research projects are still a conceptual model instead of a real physical design. The gap between the bond graph and the actual configuration owes to the fact that there is no direct mapping from the bond graph elements to physical components. In our approach, we start from the physical topology instead upon which bond graph models are aggregated computationally.

Any component may be modeled with various levels of complexity or fidelity. The selection of the appropriate level of complexity of a model is critical to a successful design experience. Modeling is considered as an art since the required complexity is not obvious in many cases. Several authors [Stein & Wilson, 1992; Stein & Louca, 1995;

Castillo & Melin, 1999; Sacks, et al, 1993] have looked at this problem and addressed the issue by deducing the most appropriate model to meet frequency domain specifications.

Similar to the research in automated modeling is the research in automated design of electro-mechanical configurations. The A-Design system developed by Campbell [2000] automated the designing of configurations by employing design agents that add elements to the design from a component library until the design meets the specified qualitative goals. This work as well as earlier approaches by Finger and Rinderle [1989], Welch and Dixon [1994] and Ulrich and Seering [1989], acknowledged the possible use of bond graphs to capture the behavior of various components. Such approaches however have not made the distinction between the functional or purpose-driven reasoning used by designers to create a configuration, and the behavioral or dynamic representation extracted in analyzing completed configurations. In the current repository efforts, the former description of function has been captured. The addition of behavioral models and design constraints within this research seeks to further provide engineering designers with a resourceful "computational design partner."

4.3 CONCEPTUAL DYNAMICS GRAPH

An appropriate model can be developed for a design configuration only if this configuration can be described in a perceivable manner. Various physical couplers can have the same functionality although they may have very different geometry, assembly methods, number of parts, etc. For example, screws, or glue can both be used to assemble structurally two components together although in different manners. Further, when we say that an automobile's engine is connected to the automobile's chassis, we are being vague. In fact, it is more precise to say that the engine mounted on the chassis is fixed to pick-up points on the engine block by means of some fasteners. That is, the relation between the engine and the chassis is most concretely described in terms of parts (or even features of parts) of the overall assembly. This is essentially a structural view of the couplings. During the conceptual design stage, the information generally needed to specify the couplings is functional in nature, rather than structural: the function of the connection between an automobile engine and a chassis is to secure the engine to the automobile's main structural support, transmit forces developed by the engine to that structure, and ensure other connections (e.g. that between the engine and the transmission) are maintained. In this situation, a Virtual Coupler (VC) is designed to capture the coupling's functional features instead of describing the structure detail.

4.3.1 Virtual Coupler

Each mechanical connection can be decomposed into couplings of 6 domains, which are three translational {x, y, z} and three rotational {th_x, th_y, th_z}. Two additional domains are included as well; electrical and hydraulic. Table 4-1 shows these 8

Table 4-1: Domain types.

Domain type	Index (D_i)
Mechanical translational *x*	D1
Mechanical translational *y*	D2
Mechanical translational *z*	D3
Mechanical rotational th_x	D4
Mechanical rotational th_y	D5
Mechanical rotational th_z	D6
Electrical	D7
Hydraulic	D8

domains (D1-D8) while the list can be extended to other domains, such as thermal, magnetic, chemical, etc. The coupling type of a specific domain can also be any of the 3 basic coupling types (Table 4-2) which are: C0 (decoupling), C1 (plain coupling) and C2 (impedance coupling). Here, plain coupling means the coupled ends at a certain domain share the same generalized flow (velocity for mechanical components).

Table 4-2: Coupling types.

Couple type	Index	Modeling Interpretation
Decoupling	C0	Disconnected ports
Plain coupling	C1	Connected directly
Impedance coupling	C2	Add Impedance element
Discrete plain coupling	C1*	Add Switch element
Discrete impedance coupling	C2*	Add switch and impedance elements

Impedance coupling implies that there exists an impedance element in the corresponding domain between the coupled two ends, which could be a resistance due to the relative motion of the two ends (e.g. the lubrication fluid of journal bearing in mechanical domains) or compliance due to the energy storage of the in-between material. Although not realized thus far, the coupling types could be augmented to account for other situations such as the 'switch' or 'ratchet' effect for a non-continuous coupling (C1* & C2*). A virtual coupler specifies how the multi-dimensional interactions between components occur as depicted in Figure 4-4.

A graph structure (Figure 4-5) composed of virtual couplers and components is called Conceptual Dynamics Graph (CD-Graph). Examples of CD-Graph will be introduced later in this chapter. Based on the conceptual dynamics (e.g. degrees of freedom or number of states), a qualitative evaluation method of design candidates is under investigation to search for the potentially feasible designs, which in fact guide the generation process of valid CD-Graphs.

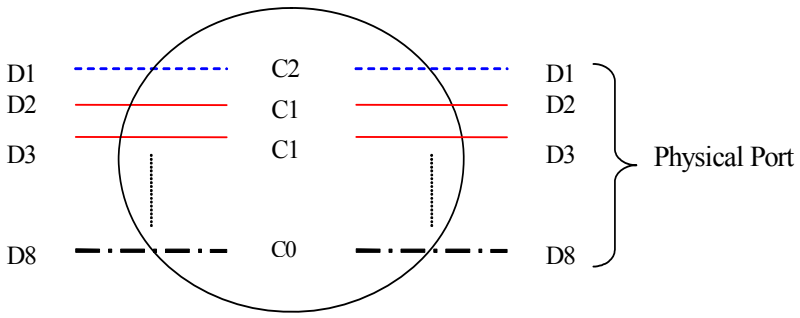


Figure 4-4: A Virtual Coupler (VC). It specifies how the interactions between components occur. It provides multi-domain information (D1 ~ D8) with certain coupling types (C0 ~ C2) at each domain between the connected components.

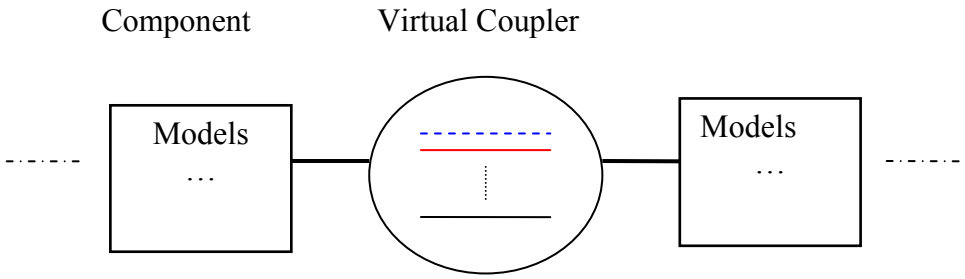


Figure 4-5: Conceptual Dynamics Graph (CD-Graph).

4.4 GENERIC MODELS

For the same component, dynamics models vary corresponding to different forms of coupling with its surroundings, which are especially true for mechanical components due to the factors of gravity and multi dimensions in the coordinate frame. To achieve the goal of modeling automation, generic models are designed for each component to fit the generic surroundings. In this section, a number of examples of generic port-based component models are demonstrated.

4.4.1 Example 1: Generic Model of a Bar

A two dimensional bar of Figure 4-6 is a standard component of mechanical systems, which can have a translational motion or a rotational motion or a combination of both. The three locations on the bar are the two tips -a and -c and a point –b in between, each of which can be used as a port to connect to another component. Each port has three domains, translational x, y and rotational th_z (θ_z). The rigid bar thus acts as a constraint between these three spatial locations [Gawthrop and Smith, 1996].

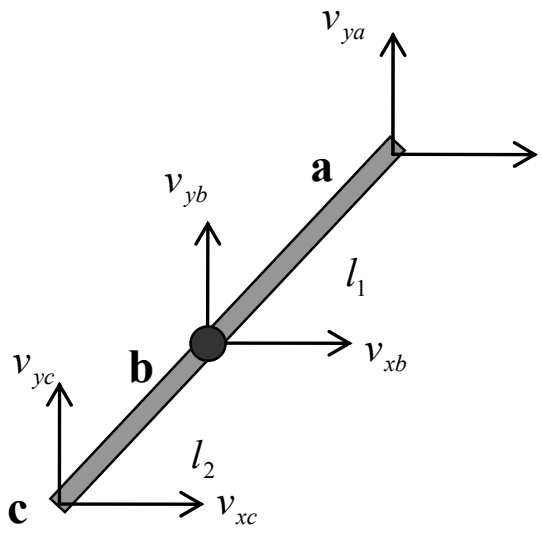


Figure 4-6: A rigid bar.

Motion is considered with respect to an absolute coordinate system: v_{xa} and v_{ya} are the components of the velocities of tip (port) -a with respect to this coordinate system; v_{xc} and v_{yc} are the components of the velocities of the tip -c; v_{xb} and v_{yb} are the velocity of the port -b in the middle. These three locations share the same angular velocity $\dot{\theta}$. The distance from the tip -a to the port -b is l_1, and the distance from the tip -c to port -b is l_2.

The kinematics of the bar are expressed by the equations

$$\begin{aligned} v_{xb} &= v_{xa} - v_{xa\_\theta} \\ v_{yb} &= v_{ya} + v_{ya\_\theta} \\ v_{xc} &= v_{xb} - v_{xc\_\theta} \\ v_{yc} &= v_{yb} + v_{yc\_\theta} \end{aligned} \qquad (4\text{-}1)$$

Where $v_{xa\_\theta}, v_{ya\_\theta}, v_{xc\_\theta}, v_{yc\_\theta}$ are the velocity components due to the angular velocity $\dot{\theta}$ given by the transformation equations

$$\begin{aligned} v_{xa\_\theta} &= l_1(\cos\theta)\dot{\theta} \\ v_{ya\_\theta} &= l_1(\sin\theta)\dot{\theta} \\ v_{xc\_\theta} &= l_2(\cos\theta)\dot{\theta} \\ v_{yc\_\theta} &= l_2(\sin\theta)\dot{\theta} \end{aligned} \tag{4-2}$$

The dynamics of the bar are given by the three (Newton-Euler) equations

$$\begin{aligned} \dot{v}_{xb} &= \frac{\Delta f_x}{m} \\ \dot{v}_{yb} &= \frac{\Delta f_y}{m} \\ \ddot{\theta} &= \frac{\Delta \tau}{J} \end{aligned} \tag{4-3}$$

Where m is the mass of the bar, J is the moment of inertia of the bar with regard to the port b. Δf_x and Δf_y are the net forces acting in the x and y directions at the port b and $\Delta\tau$ is the net torque acting at the port -b.

The corresponding bond graph appears in Figure 4-7. The source sensors (SS) indicate how the connections are made to the bar. The bulleted list below explains in detail how this bond graph is mapped to the equations.

- There are three I components labeled ‘m_x’, ‘m_y’, and ‘J’; these implement the three dynamic Equations (4.3).
- There is one integrated transformer (INTF) component to take angular velocity $\dot{\theta}$ as the input and give the integrated flow q as output.

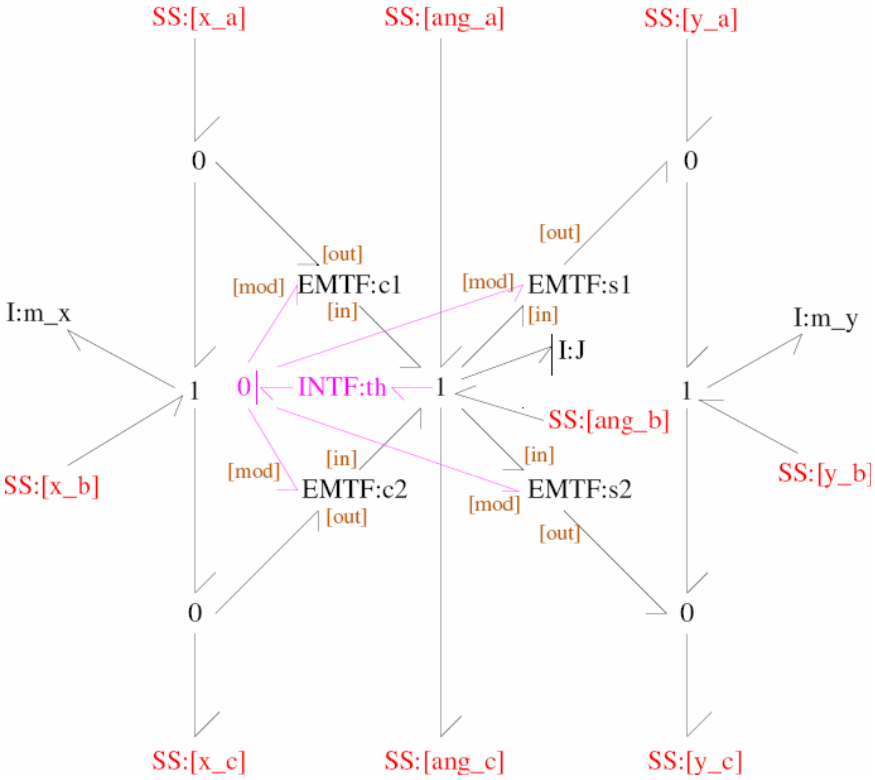


Figure 4-7: Generic model of a 2D 3-port Bar.

$$q = \int_0^{t'} \dot{\theta}(t)dt = \theta + q_0 \tag{4-4}$$

If initialized in such a way that $q_0 = 0$, then $q = \theta$, thus provides a modulating signal for the effort modulated transformers (EMTF).

- There are three 1-junctions that carry the three velocities associated with the port --b: v_{xb}, v_{yb} and $\dot{\theta}$. These three 1-junctions each compute the net effort acting on the corresponding I element (Δf_x, Δf_y and $\Delta\tau$).

- There are four 0-junctions that carry the x and components of the force associated with the port -a and -b of the bar. These four 0-junctions imply the four kinematic Equations (4.1).
- There are four transformers labeled ‘c1’, ‘s1’, ‘c2’, ‘s2’.
    1. These four transformers imply the four transformation Equations (4.2).
    2. These four transformers, by power conservation, also imply the corresponding force transformations.
    3. These four transformers are each modulated by θ that is generated by the integrated transformer (INTF) labeled as ‘th’.

Note that the model of a mechanical lever could be easily obtained from this bar model by pinning any of the 3 ports to a fulcrum. Further, more ports can be added in the middle of the bar in the same manner as the end ports.

4.4.2 Example 2: Generic Model of a Spring

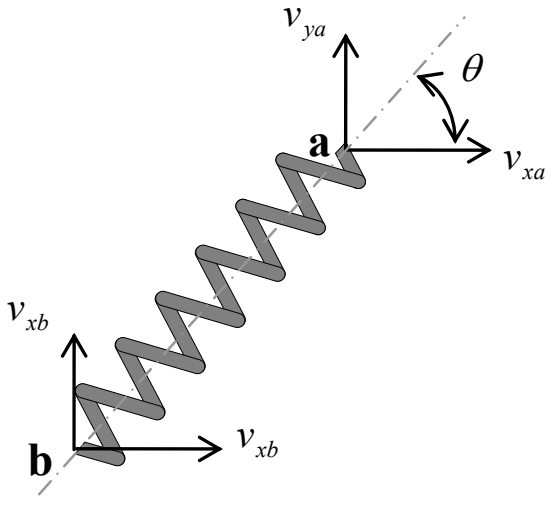


Figure 4-8: A spring with port -a and port -b

The spring of Figure 4-8, moving in the plane, is a standard component of mechanical systems. A translational spring within a design configuration can be simply modeled as a linear capacitance if the motion of its two ends remains along the same line. However, if the spring exhibits motions that are planar (2-D) or even out of plane (3-D), a more complicated model is needed.

The two locations on the spring are the two tips -a and –b, each of which can be used as a port to connect to another component. The spring thus asserts a force against the components connected through these two ports. Motion is considered with respect to an absolute coordinate system: v_{xa} and v_{ya} are the components of the velocities of tip (port) -a with respect to this coordinate system; v_{xb} and v_{yb} are the components of the velocities of the tip –b. The distance from the tip –a to the tip -b is represented as l, and with the distance at zero spring load as l_0.

The kinematics of the spring are expressed by the equations:

$$\begin{aligned} \dot{l}_x(t) &= v_{xa} - v_{xb} \\ \dot{l}_y(t) &= v_{ya} - v_{yb} \end{aligned} \tag{4-5}$$

Where $\dot{l}_x(t), \dot{l}_y(t)$ are the rates of length change along the x and y direction.

$$\begin{aligned} l_x &= \int_0^{t'} \dot{l}_x(t)dt + l_{x0} \\ l_y &= \int_0^{t'} \dot{l}_y(t)dt + l_{y0} \end{aligned} \tag{4-6}$$

Where l_x, l_y are the spring length on the x and y direction with l_{x0}, l_{y0} as the initial conditions. The position angle of the spring relative to the reference frame can be obtained by:

$$\dot{\theta} = \arctan(\frac{l_y}{l_x}) \tag{4-7}$$

$$\theta = \int_0^{t'} \dot{\theta}(t)dt + \theta_0 \tag{4-8}$$

Further,

$$\begin{aligned} \dot{l}_{x\theta} &= \dot{l}_x \cos(\theta) \\ \dot{l}_{y\theta} &= \dot{l}_y \sin(\theta) \end{aligned} \tag{4-9}$$

Where $\dot{l}_{x\theta}$ and $\dot{l}_{y\theta}$ are the projections of $\dot{l}_x$ and $\dot{l}_y$ along the spring axis. The addition of $\dot{l}_{x\theta}$ and $\dot{l}_{y\theta}$ gives the rate of change of the spring length as following:

$$\dot{l} = \dot{l}_{x\theta} + \dot{l}_{y\theta} \tag{4-10}$$

The dynamics Equations is simply:

$$\Delta f = f_c(\int_0^{t'} \dot{l}(t)dt + l_0) \tag{4-11}$$

Where Δf is the net force acting between the port –a and -b. If we assume this is an ideal linear spring with C as the spring compliance,

$$\Delta f = \frac{\int_0^{t'} \dot{l}(t)dt + l_0}{C} \tag{4-12}$$

The corresponding bond graph appears in Figure 4-9. The source sensors (SS) indicate a spring's domain connections to other components.

- There are two -0 elements at the most left and right of the bond graph to calculate the rates of change of the spring length along the x and y directions.

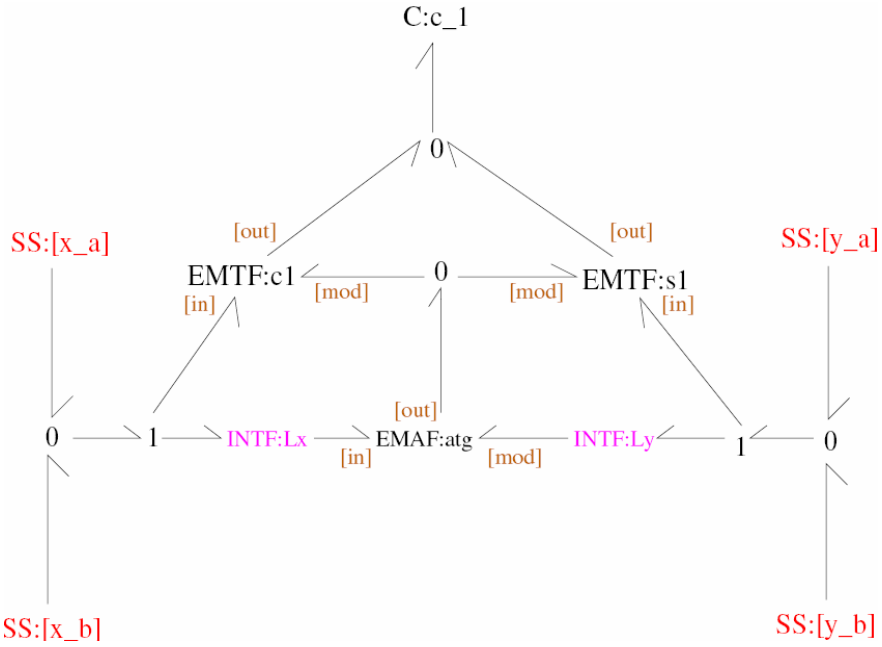


Figure 4-9: Generic model of a 2-port Spring.

- There are two integrated transformers (INTF) labeled as 'Lx' and 'Ly' that take the change rates of the spring length along x and y directions as inputs and give the spring length along x and y directions as outputs respectively through integration as in Equations (4.6).
- There is one effort modulated amplifier (EMAE) labeled as 'atg' to calculate the angle θ as in Equation (4.7).

- There are two effort modulated transformers (EMTF) labeled as ‘c1’ and ‘s1’ to imply Equations (4.9) that both use angle θ as the modulator (an effort variable shared by the bonds connected to the -0 element in the middle of the bond graph).
- The -0 element on the very top of the bond graph implies Equation (4.10).

The state space equations generated from this bond graph is shown in Figure 4-10.

$$\dot{x}_1 = \cos(z_1) * u_1 + \cos(z_1) * u_2 + \sin(z_1) * u_3 + \sin(z_1) * u_4$$

$$\dot{x}_2 = u_1 + u_2 \qquad \dot{x}_3 = u_3 + u_4$$

$$z_1 = \arctan(\frac{x_2}{x_3})$$

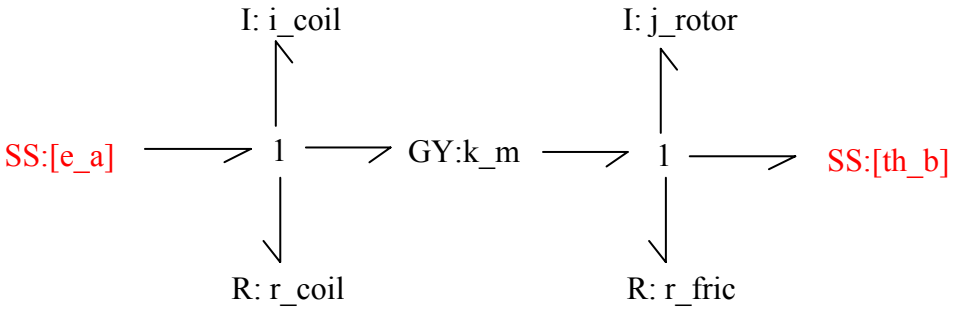
$$y_1 = \frac{\cos(z_1) * x_1}{c1} \qquad y_2 = \frac{\cos(z_1) * x_1}{c1}$$

$$y_3 = \frac{\sin(z_1) * x_1}{c1} \qquad y_4 = \frac{\sin(z_1) * x_1}{c1}$$

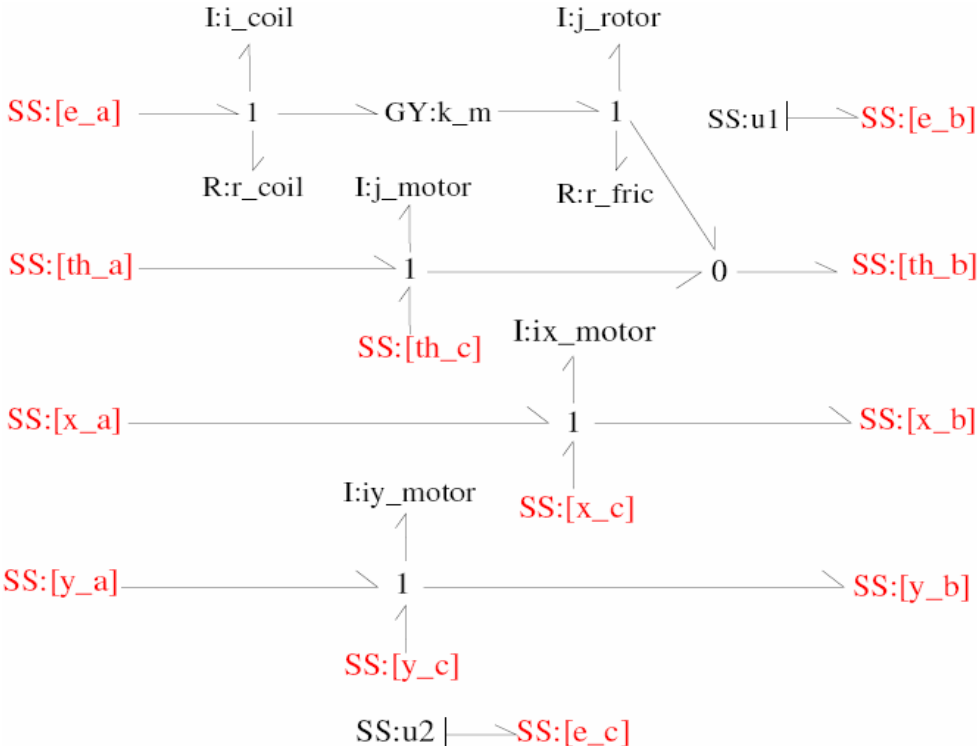
Figure 4-10: Differential algebraic equations of the 2-port Spring

4.4.3 Example 3: Generic Model of a Motor

Figure 4-11 (a) shows a simple bond graph motor model. This model underlines an assumption that the motor is grounded and the inertia of motor body can be ignored, which does not hold when the motor is mounted on a movable chassis. In our application of automated model generation, coupling patterns with surroundings need to be understood autonomously. Figure 4-11 (b) shows a generic motor model that has three ports, one for an electrical connection (port a), one for a rotational output (port b), and another one for the motor base mounting (port c). There are 4 domains considered for each port, including one electrical domain (indicated in the figure as SS:[e_a]) and three



(a): A 2-port Motor model.



(b): Generic model of a 3-port Motor.

Figure 4-11: Motor models

mechanical domains. Note that some dangling source sensors (e_b and e_c) exist in the bond graph, which are deployed to apply zero flow (specified in SS:u1 and SS:u2) to the corresponding domain of whatever components are to be coupled with these ports. Note that for each component model, following the model structure of MTT, there is another label text file associated with its bond graph to specify the detail of its bond graph elements (e.g., the zero flow information of SS:u1).

4.4.4 Example 4: Generic Model of a Wheel (Gear)

In this section, generic models for a wheel (without considering compliance) or a gear will be derived starting from the consideration of general rigid body model, which facilitates systematic model derivation of a specific component.

4.4.4.1 Dynamics of a general rigid body

Figure 4-12 shows a general rigid body both translating and rotating in space. Inertial axes X, Y, Z are shown, and axes x, y, z are attached to the body, at its center of mass, and aligned with the principal axes of the body. With respect to these body-fixed coordinates, the rotational inertial properties remain invariant and the products of inertia are all zero. While these body-fixed coordinates are not the best from which to view the body motion, they are practical coordinates for the computation of body motion. At the instant shown, the body has absolute velocity $\vec{v}$ and absolute angular velocity $\vec{w}$. These vectors have been cast into three mutually perpendicular components: v_x, v_y, v_z and w_x, w_y, w_z. Newton stated that the net force F acting on the body changes its momentum:

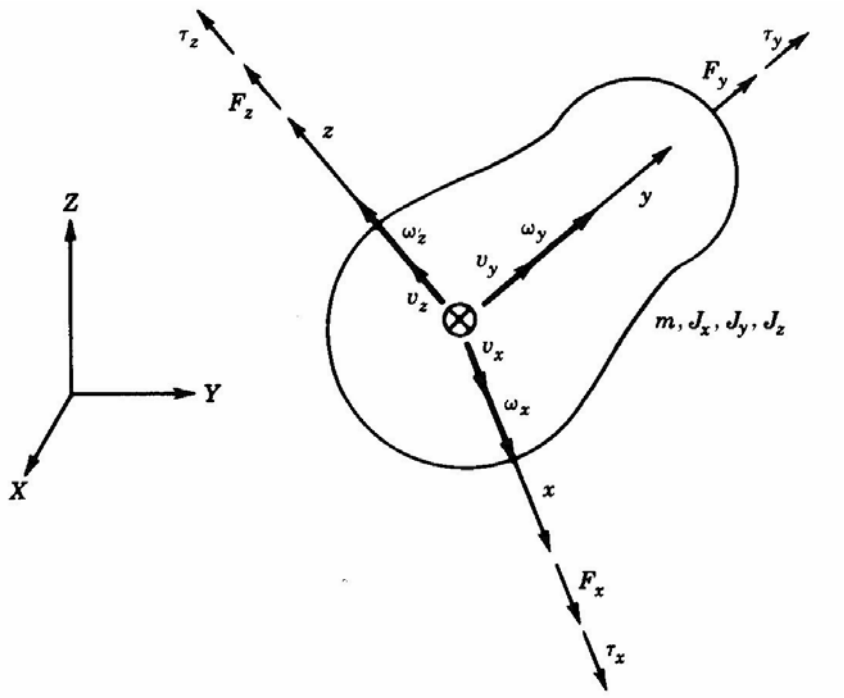


Figure 4-12: Body in general 3-D motion. [Karnopp et al. 2000]

$$F = \frac{d}{dt} p \tag{4-13}$$

Where

$$p = m\vec{v} \tag{4-14}$$

If V is expressed with respect to a rotating frame, then

$$F = \dot{p} + \vec{w} \times p \tag{4-15}$$

In the same fashion,

$$\tau = \dot{h} + \vec{w} \times h \tag{4-16}$$

$$h = J\vec{w} \tag{4-17}$$

Here τ is the net torque acting on the body, h is the angular momentum with respect to the center of mass, and J is a diagonal matrix of the principal moments of inertia.

Following the rule for vector product, we have:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix} + \begin{bmatrix} 0 & p_z & -p_y \\ -p_z & 0 & p_x \\ p_y & -p_x & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{4-18}$$

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} J_x \dot{w}_x \\ J_y \dot{w}_y \\ J_z \dot{w}_z \end{bmatrix} + \begin{bmatrix} 0 & h_z & -h_y \\ -h_z & 0 & h_x \\ h_y & -h_x & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{4-19}$$

4.4.4.2 Coordinate Transformations

Since it is unlikely that external forces and torques will be conveniently lined up with the continuously changing principal directions, and since the body motion is difficult to interpret with respect to the body-fixed coordinates, it is necessary to transfer from body-fixed coordinated to other more convenient frames through a series of Cardan angle coordinate transformations.

Cardan angles are to be introduced that correspond to the yaw, pitch and roll angles of an automobile. Figure 4-13 shows an inertial frame (X,Y,Z), a body fixed orientation (x,y,z), and two intermediate frames (x',y',z') and (x'',y'',z''). The body-fixed frame is arrived at by first rotating about Z through angle ψ (yaw), yielding the frame (x',y',z'). We next rotate about the y'-axis through the angle θ (pitch), yielding

the $x", y", z"$ axes. Finally, we rotate about the $x"$-axis through the angle ϕ (roll), yielding the instantaneous body-fixed frame, x, y, z.

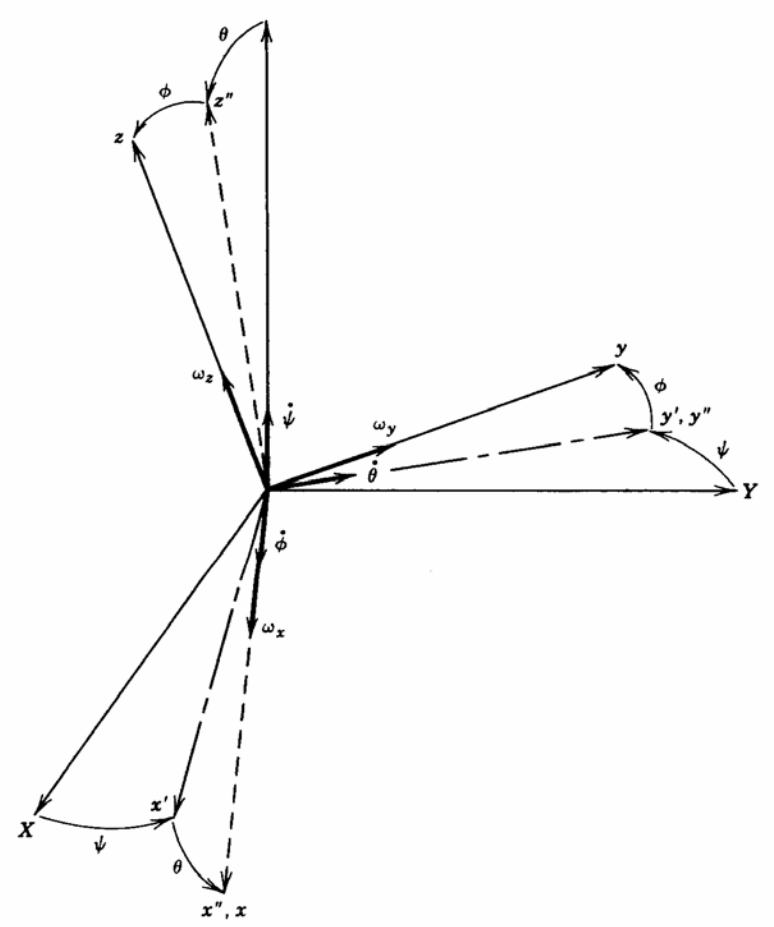


Figure 4-13: Cardan angle coordinate transformation. [Karnopp et al. 2000]

From these rotations, we can write out relationships in matrix form as:

$$\begin{bmatrix} w_{x"} \\ w_{y"} \\ w_{z"} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{4-20}$$

$$\begin{bmatrix} w_{x'} \\ w_{y'} \\ w_{z'} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} w_{x''} \\ w_{y''} \\ w_{z''} \end{bmatrix} \tag{4-21}$$

$$\begin{bmatrix} w_X \\ w_Y \\ w_Z \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_{x'} \\ w_{y'} \\ w_{z'} \end{bmatrix} \tag{4-22}$$

Thus, if we know w_x, w_y, w_z, then we can determine the angular velocity components in all frames, including the inertial frame. Let:

$$\Phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{4-23}$$

$$\Theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4-24}$$

$$\Psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-25}$$

Then we will have

$$\begin{bmatrix} w_X \\ w_Y \\ w_Z \end{bmatrix} = \Psi\Theta\Phi \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{4-26}$$

In the same fashion

$$\begin{bmatrix} v_X \\ v_Y \\ v_Z \end{bmatrix} = \Psi\Theta\Phi \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{4-27}$$

According to power conservation, the relations among the forces and torques can be derived as:

$$\begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} = (\Psi\Theta\Phi)^t \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{4-28}$$

$$\begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_Z \end{bmatrix} = (\Psi\Theta\Phi)^t \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{4-29}$$

Note that the angles ϕ, θ and ψ are needed to modulate the transformation.

It is relatively simple to relate $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ to the body-fixed components w_x, w_y, w_z from Figure 4-13.

$$w_x = \dot{\phi} - \dot{\psi}\sin\theta \tag{4-30}$$

$$w_y = \dot{\theta}\cos\phi + \dot{\psi}\cos\theta\sin\phi \tag{4-31}$$

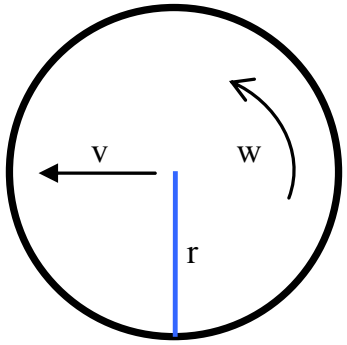
$$w_z = -\dot{\theta}\sin\phi + \dot{\psi}\cos\theta\cos\phi \tag{4-32}$$

Easy to solve the equation to get

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \\ 1 & \sin\phi\sin\theta/\cos\theta & \cos\phi\sin\theta/\cos\theta \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{4-33}$$

4.4.4.3 Wheel (Gear) Models

The general 3-D rigid-body dynamics model is presented in more details in [Karnopp et al. 2000]. Based on this general model, a 3-D wheel can be modeled by adding extra constraints.



$v = w * r$

Figure 4-14: Wheel in 2-D motion.

A 2-D wheel is first shown in Figure 4-14, which has only rolling motion. The relative velocity between the outer contour and the wheel center (or the absolute velocity of disc center if no slipping between the wheel and the ground), is the product of the rolling angular velocity and the radius of the wheel.

$$v = w * r \tag{4-34}$$

There is a similar rule for the wheel moving in 3-D space if we pay attention to the body fixed coordinates as shown in Figure 4-15.

$$\vec{v} = \vec{w} \times \vec{r} \tag{4-35}$$

It is obvious that, relative to the body fixed coordinates, $\vec{r}$ is independent of the pitch angle θ and the yaw angle ψ, but dependant on ϕ, the rolling angle along x axis, which is:

$$\vec{r} = (0, R\sin\phi, R\cos\phi) \tag{4-36}$$

Following the rule for vector product, we have:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 0 & R\cos\phi & -R\sin\phi \\ -R\cos\phi & 0 & 0 \\ R\sin\phi & 0 & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{4-37}$$

$$\text{Let} \quad T = f(\phi) = \begin{bmatrix} 0 & R\cos\phi & -R\sin\phi \\ -R\cos\phi & 0 & 0 \\ R\sin\phi & 0 & 0 \end{bmatrix} \tag{4-38}$$

This matrix tells how the angular velocity vector can be related to linear velocity vector in the body-fixed coordinate frame originated at center point of the wheel as shown in Figure 4-15.

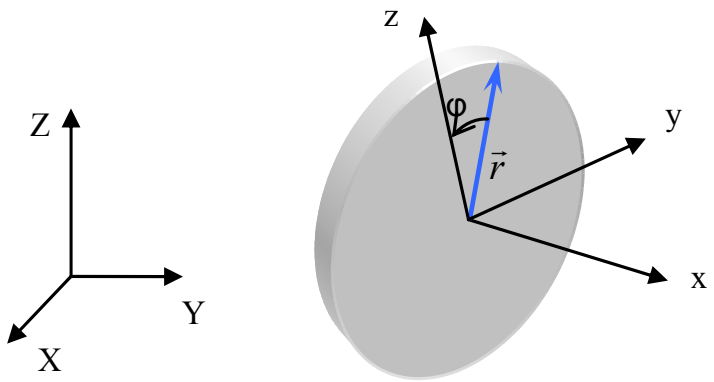


Figure 4-15: Wheel in 3-D motion

There are different levels of generic models for a component in terms of the number of ports and number of domains for each port. From equations derived in this section, we can have the 2D generic wheel bond graph model as in Figure 4-16 and 3D generic wheel bond graph (ignoring compliance) as in Figure 4-17 (same for a gear). The models with different complexity have the same two ports, one (port a) is the center of the wheel and the other

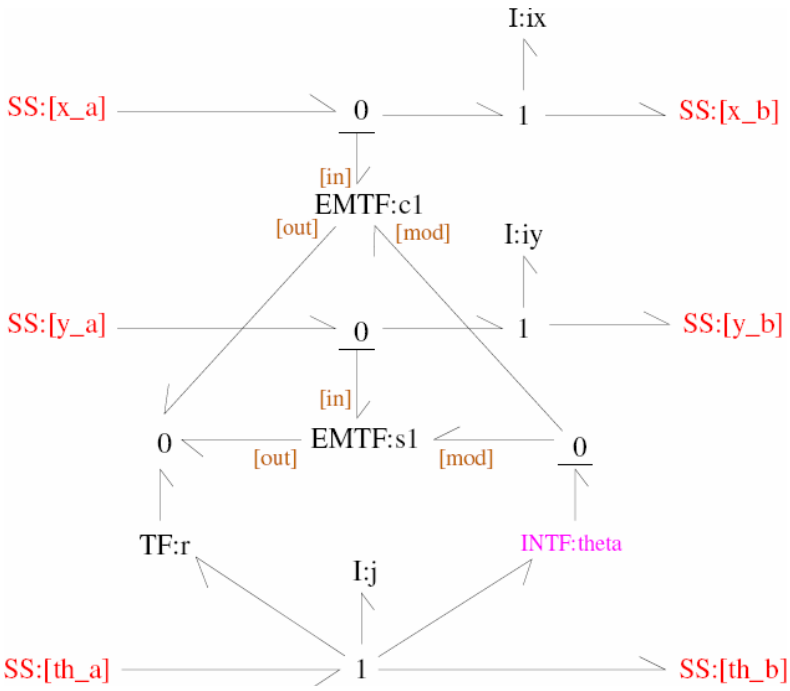


Figure 4-16: 2D generic model of a Wheel (Gear) with 2 ports.

(port b) is the outer contour. The same generality applies that either port (domain) can be, in practice, -- rigidly coupled, slipping, or decoupled. The 3D model has 6 domains instead of 3 domains for each port and is more complicated.

The famous Euler Disk problem [Easwar et al., 2002] as shown in Figure 4-18 (a) can be model easily from the generic 3D model by connecting to the ground using a virtual coupler (Figure 4-18 (b)) that has the translational-z domain rigid coupled and all the other domains decoupled. The simulation of an Euler disk in Figure 4-19 shows the motion path of the disk center in XY plane and the height of the disk center in function of time. In this simulation, air dissipation and ground friction are neglected intentionally. As shown in Figure 4-19 (b), the height of the disk center is keeping decreasing because of

the gravity along the negative Z direction. More simulation and the related matlab code can be found in Appendix C.

Note that, for all the demonstrated examples, coordinate transformation between component body frame and reference frame is integrated into the bond graph representation of component models. This mechanism provides a fundamental convenience to generate the system model.

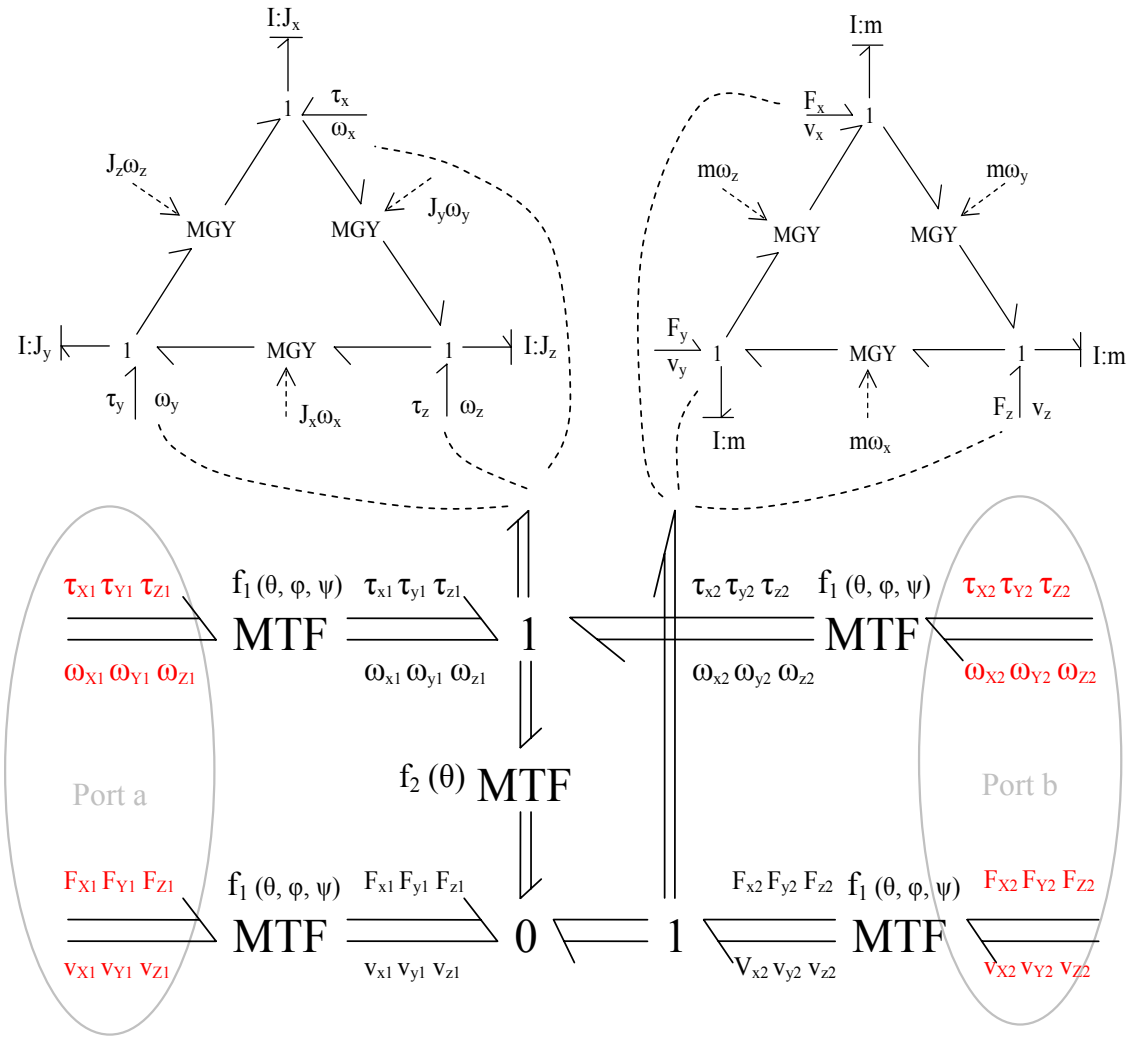


Figure 4-17: 3D generic model of a Wheel (Gear) with 2 ports.

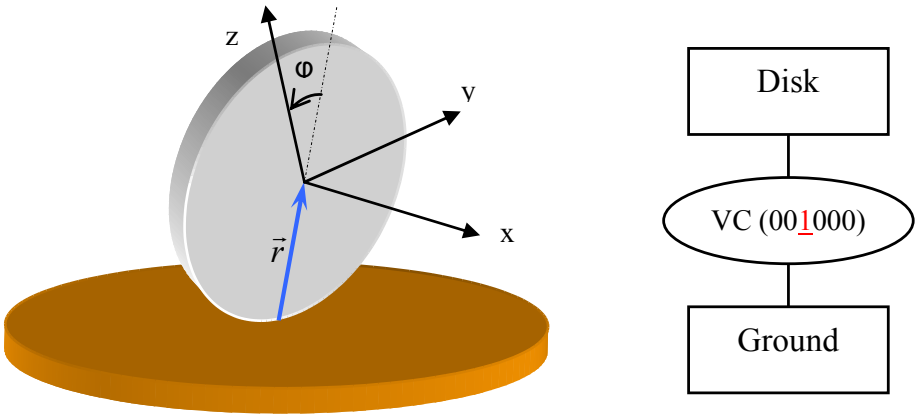
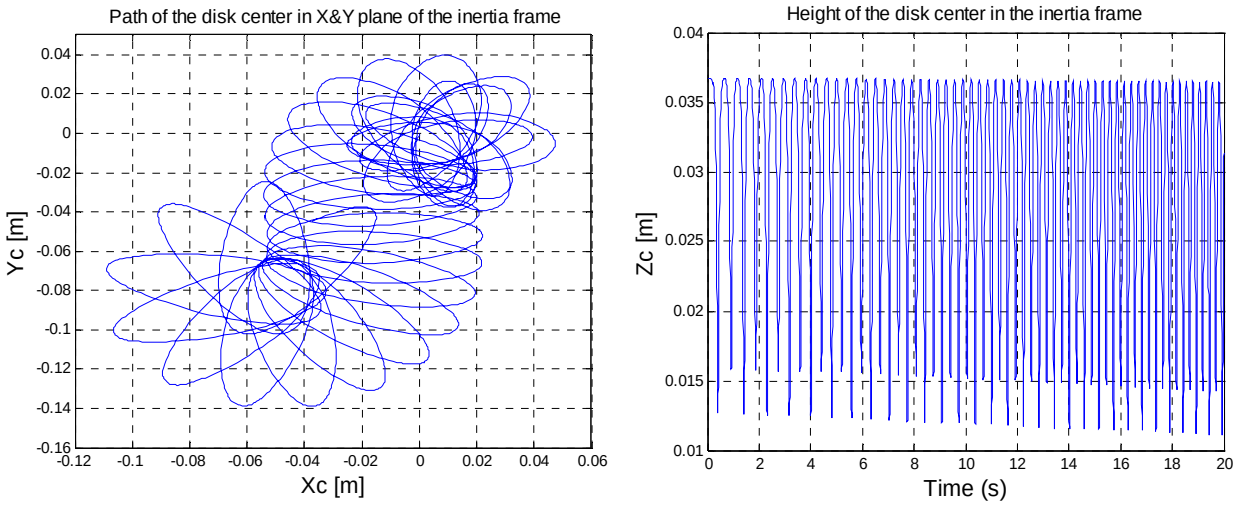


Figure 4-18: Euler Disk and its CD Graph. The disk and the ground are rigidly coupled indicated as '1' while all other domains are decoupled as '0'.



(a) Path of disk center in XY plane (b) Height of disk center in Z direction

Figure 4-19: Simulation of Euler disk free spinning on the ground

Parameters:	ρ = 7830 (kg/m^3	Density of steel
	r = 0.0367 (m)	Radius of disk
	t = 0.0127 (m)	Thickness of disk
Initial Conditions:	Wx0 = pi/8 (rad/s)	roll angular velocity
	Wz0 = 2*pi (rad/s)	pitch angular velocity

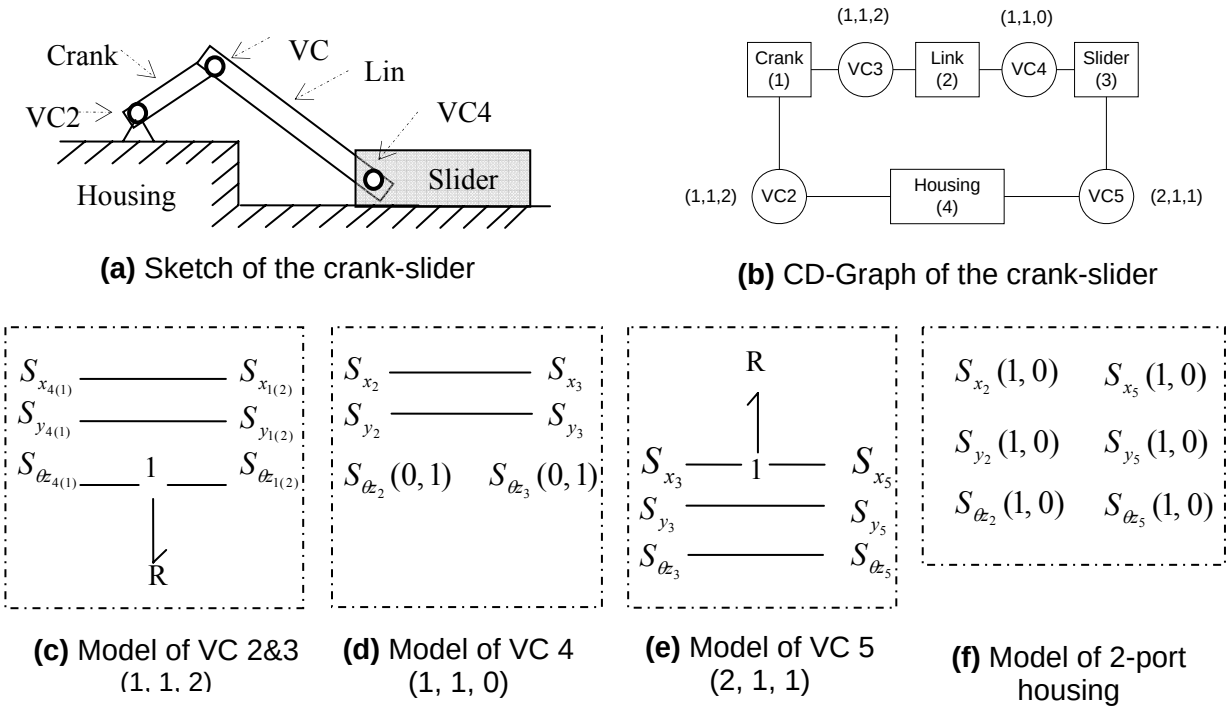
4.5 SYSTEM MODEL GENERATION

Based on bond graphs, the CD-Graph provides a "design" representation that facilitates the automated modeling of multi-domain design configurations through a mapping between a system's generic model configuration and physical topology configuration. This mapping also indicates that when a physical system is represented in a Cartesian coordinate frame, so can the model(s) of the physical system.

A schematic 2D crank-slider design structure is shown in Figure 4-20 (a) as an illustrative example of system level model generation. Figure 4-20 (b) shows the CD-Graph of this design configuration. Each 2D-port of the components and virtual couplers has three domains (x, y, θ_z), with all the other mechanical domains plain coupled and non-mechanical domains decoupled. Figure 4-20 (c), (d) and (e) represent the bond graph models for virtual couplers VC2 & VC3 (C1, C1, C2), VC4 (C1, C1, C0), and VC5 (C2, C1, C1) respectively. VCs 2&3 has plain coupling for x- & y-domain (marked as C1) and impedance couple for θ_z-domain (marked as C2). VC4 is different from VCs 2&3 since its θ_z-domain is decoupled by the assumption that there is no rotational friction (marked as C0). VC5 has its x-domain defined as impedance coupled (marked as C2).

Note in the expression $S_x(0, 1)$ in (d), where S_x is the model interface corresponding to the x-domain of the physical port (following the definition of MTT from [Gawthrop and Bevan, 2003]). Within the parenthesis following S_x, the first binary digit represents effort (force in mechanical), and the second one represents flow (speed in mechanical). $S_x(0, 1)$ means that at x-domain effort variable is grounded to zero, while x-domain flow variable is determined by dynamics of the overall system.

Visually, in the system's bond graph, S(0,1) works as a "plug" (or a single domain component) in a domain, which asserts a free end (zero effort) to the corresponding domain of the connected component. For a 2D pendulum example, the free end is coupled with a virtual coupler modeled simply by three S(1,0) plugs since this end is free of restriction at any of the 3 domains.



(a) Sketch of the crank-slider

(b) CD-Graph of the crank-slider

(c) Model of VC 2&3 (1, 1, 2)

(d) Model of VC 4 (1, 1, 0)

(e) Model of VC 5 (2, 1, 1)

(f) Model of 2-port housing

Figure 4-20: The *system model generation* of a 2-D crank-and-slider system. This task can be done by simply connecting the ports in a domain-to-domain manner. The schematic geometric expression of the simplified crank-slider conceptual design is shown in (a) and its CD-Graph in (b). Each port of the components and virtual couplers has three domains (*x, y,* θ_z), with all the other mechanical domains plain coupled and non-mechanical domains decoupled. Figures (c), (d) and (e) represent the bond graph models for virtual couplers VC2 & VC3 (C1, C1, C2), VC4 (C1, C1, C0), and VC5 (C2, C1, C1) respectively. A 2-port ground model is shown in (f), although it can be expanded easily to as many ports as needed. The bond graph model for crank and link (both simplified as bar model) is shown in Figure 4-7.

Further, a 2-port ground model is shown in (f), although it can be reduced to 1 port or expanded to as many ports as needed. For this ground model all the domains have zero flow but an arbitrary value for effort which is determined by the input, so the x-domain of ground component can be marked as $S_x(1, 0)$, likewise for the other two domains. Visually S(1,0) works as a "plug" in a domain, which assert a ground (zero flow) to the corresponding domain of the connected component. This feature becomes useful when a simple model of a component (e.g., a 2D model while the other domains are grounded) is connected to a complicated model of another component (e.g., a 3D model). Any extra domain of the virtual coupler (one end 6 domains, another end 3 domains) between the two models can be connected to (or plugged by) a single domain component that is simply modeled as S(1,0).

Given the general behavior models of all the components (Both crank and link are considered as a bar model as shown in Figure 4-7), and all the virtual coupler models, a system level model (Figure 4-21) can be created by simply connecting the components and couplers intuitively in a domain-to-domain manner (i.e. x to x, y to y).

After the system bond graph model is generated, ground information can be applied to remove the bond graph elements with no energy effect on the rest of the system (such as the dangling elements). There are other energy-based bond graph model reduction algorithms ([Sendur et al. 2003], [Louca and Stein, 1999]), but all with the condition that the detailed design parameters are known, which is not true for our application at the conceptual design stage.

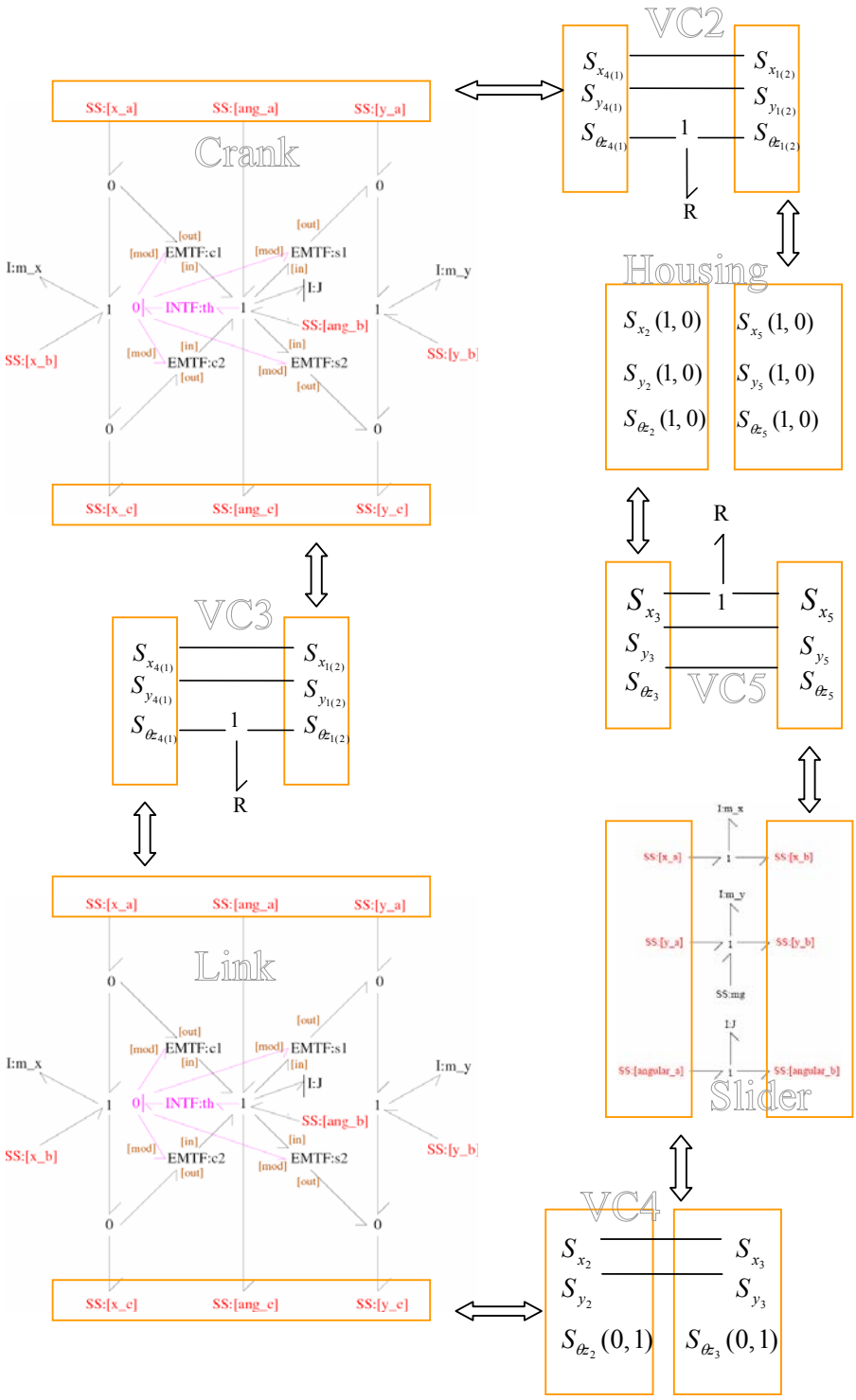


Figure 4-21: The system bond graph model generated from CD-Graph.

4.6 AUTOMATED PREPARATION OF GENOTYPES FOR OPTIMIZATION

After a system dynamics model (Bond Graph) is generated, the model transformation from this graphical representation to other representations such as state space or transfer function can be automated through the Model Transformation Tool (MTT) [Gawthrop, 1995]. Design goals (e.g. settling time for a step input, overshoot, bandwidth, etc.) are often as clearly defined as possible by engineers prior to any design effort, which suggests that the evaluation routine for designs can be obtained by representing the design goals as a function of system outputs variables, which are derivable from the system inputs and dynamics model. Based on this evaluation routine, an optimization task can be invoked to determine the choices for the design variables in each of the components pulled from the repository into the configuration. Design variables can be decisive variables (e.g. length of a bar component) or a dependant variables (e.g. stiffness of a spring component).

The dynamics models built will likely include significant nonlinearities that results in a multi-modal design space. Since gradient-based optimization methods will most likely fail in such scenarios, genetic algorithm (GA) [Holland, 1992] – a stochastic-based approach – is adopted to find the best set of parameters. The major appeal is due to its attributes of robustness and global searching. In a GA, the design solution candidates (phenotype) are encoded in a list representation called a genotype. A unique and necessary task in this research is to, given the evaluation routine, automatically prepare a design problem for the application of optimization using genetic algorithm. This preparation sets up the genotype representation for a design by encoding design variables, while taking into consideration of the design constraints and physical

constitutive laws that were pre-specified in the component repository. An example of weighing machine design is used to showcase this research for detailed explanation.

4.7 CASE STUDY OF A WEIGHING MACHINE DESIGN

Figure 4-22 shows an example structurally of the optimization problem for a weighing machine design with its CD-Graph shown in Figure 4-23. The weighing machine conceptual design takes weight as input to a footpad that is rigidly coupled with a lever [Campbell, 2000]. A spring is used to support the lever (a bar) whose translational motion at the free end drives a rack-pinion (a bar and a gear) to generate rotational motion, which pass through a shaft and then displayed by a gauge (dial) displacement.

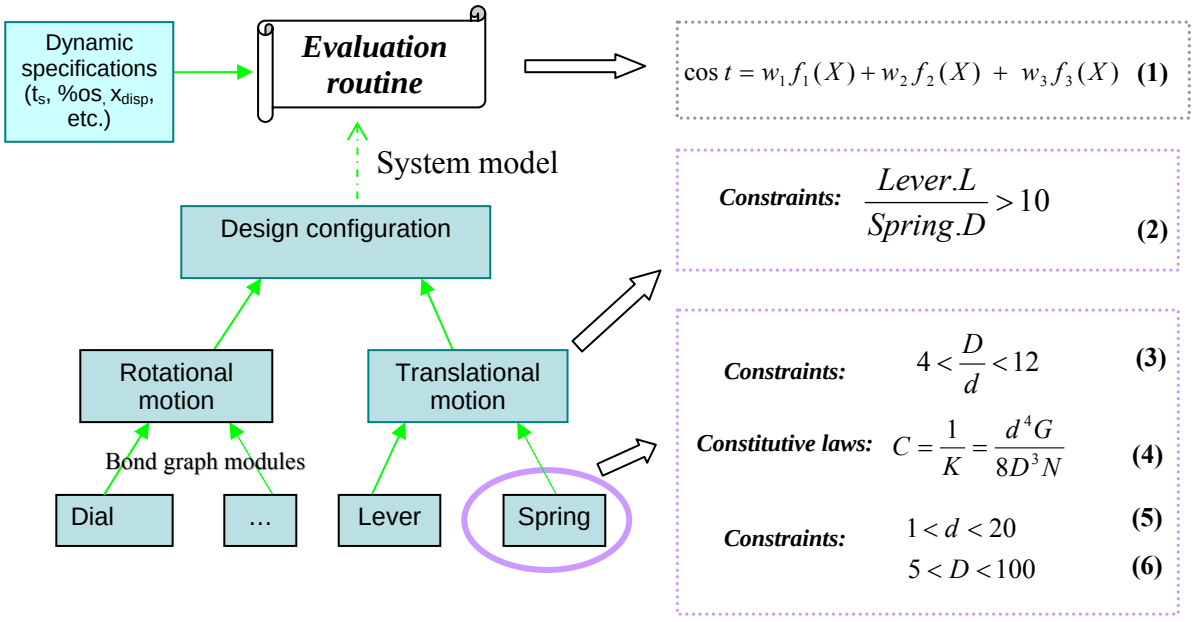


Figure 4-22: Optimization for a weighing machine design. Samples of design constraints and constitutive laws are listed at different levels, such as the traditional design rules (Eq. 2) and manufacturing constraints (Eq. 5&6). The evaluation routine of a design is pre-specified according to design goals and invokes the derived system model for the purpose optimization using genetic algorithm.

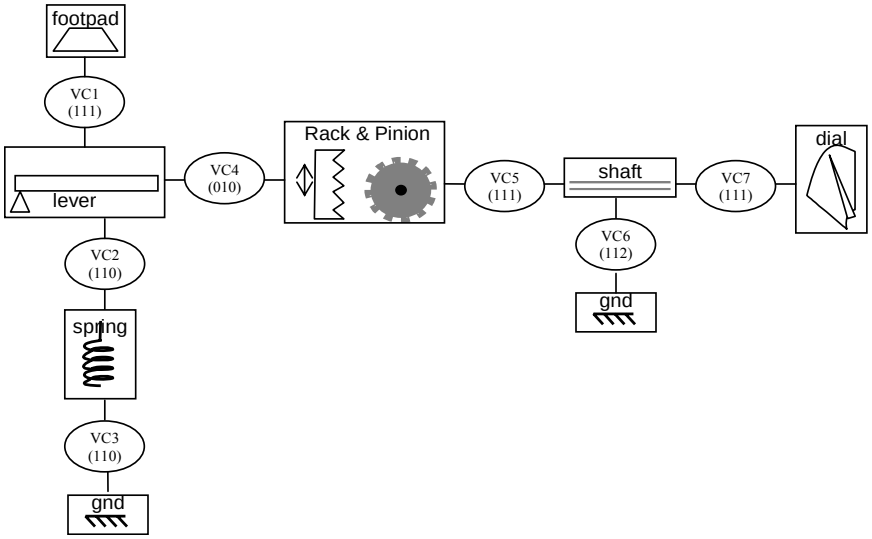


Figure 4-23: CD-Graph of a weighing machine design

Among the specifications of Figure 4-22 are the dynamics goals such as the oscillation of the output gauge is expected to settle down within three seconds and the percent overshoot is expected to be less than 20%, etc. As a result, the optimization is faced with multiple objective functions as will often be the case in such problems. The system model of this design can be automatically generated from the CD-Graph and then converted into desired formats (e.g. transfer functions between multiple inputs and outputs), which is invoked by the evaluation routine (Eq.1 of Figure 4-22) that reflect the design specifications. The computational choice of component parameters should comply with the "rules of thumb" that are followed by the design engineers. These rules may be cast as equality or inequality constraints for the optimization problem. For instance, as shown in Eq. 3 of Figure 4-22, the ratio between spring coil diameter and wire diameter is usually expected to greater than 4 to allow for easier fabrication and less than 12 to avoid tangling. Also, the constraints of system level (relation between variables from

different components) can be pre-defined or open to designer's specification. As shown in Eq. 2 of Figure 4-22, a design heuristic posed a requirement that the length of lever is at least 10 times larger than the coil diameter of the spring.

The bond graph system model is represented with the attributes of the components and couplers, such as the capacitance element C and resistance element R, which are intermediate variables from a perspective of product design. It is our goal to relate the generalized bond graph attributes of components to basic physical design variables, such as dimensions, materials, etc. Constitutive laws or mappings between the intermediate and basic variables are needed which are defined in the "behavioral model" section of the component representation (Figure 4-2). For example, Eq. 4 of Figure 4-22 defines the spring stiffness as a function of basic design parameters including the number of coils, coil diameter, etc. In practice, the physical design variables are constrained to certain boundaries, such as the dimensional range, or to some limited discrete values, such as the material properties. These constraints are defined within the "Design Constraints" section of the component representation. Eqs. 5 and 6 in Figure 4-22 are examples of such boundaries for wire diameter and coil diameter respectively of spring design.

4.7.2 Preparing Design Genotype

A unique task in this research is to form the genotype automatically, which entails: 1) derive the proper set of design variables and their range; and 2) encode design constraints into the genotype to reduce the searching space.

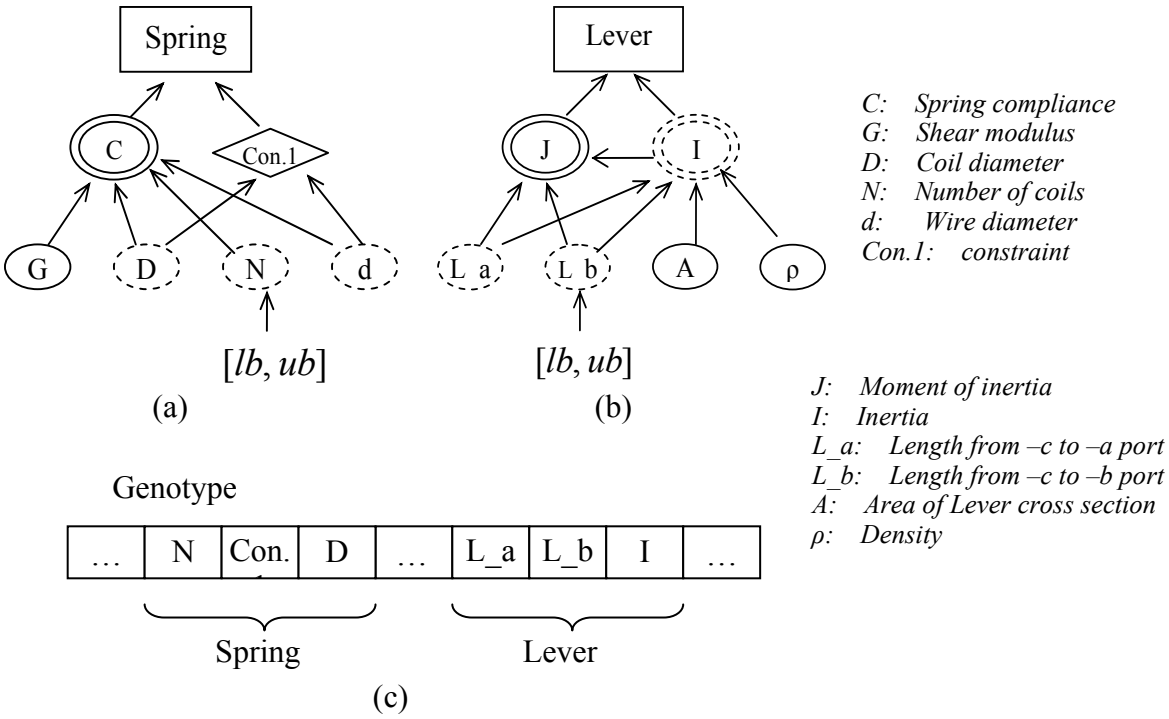


Figure 4-24: Automatic construction of genotype representation (c) based on the hierarchical representation of component design rules (a) & (b).

4.7.2.1 Design Variables

In this research, component design information such as constitutive laws and constraints are captured through a hierarchical structure (e.g., Figure 4-24 (a) & (b)) with the top node marked as component name, middle nodes as intermediate design variables (nested circles) and the leaf nodes as basic design variables (single circles). In the spring example (a), the constraint (diamond) and the constitutive relation correspond to Eqs. 4 and 5 of Figure 4-22 respectively and the lever example (b) captures the constitutive relations between J and I, L_a, L_b and between I and L_a, L_b, A, ρ.

Note that any physical parameter can be a design variable and be subject to a tuning process. However, prior to the beginning of the computational design process,

the designer has the option to fix the values of some variables (e.g., variable G marked in solid line circle(s)) and/or to set up boundaries for some other design variables (e.g., variable N in Figure 4-24 (a)). If an intermediate variable is chosen as design variable (e.g. variable I in Figure 4-24 (b)), its single-parented children (such as variable A) will not be chosen as design variables to prevent conflicts. The dashed circles (single or nested) are the potential set of design variables to be encoded in the genotype when not considering encoding design constraints.

4.7.2.2 Encoding design constraints

Instead of applying design constraints to screen design candidates for each loop of design generation, it is more efficient to embed constraints (equality and inequality) into the genotype representation. A constraint can be, in general, represented as in Eq. 1 of Figure 4-25. If a transformation can be created by expressing any design variable, say x_1, as a function of other variables and the constraint variable 'x_{con}', then the design variable set $\{x_1, x_2, \ldots x_n\}$ governed by this constraint becomes $\{x_{con}, x_2, \ldots x_n\}$as in Eq. 2 of Figure 4-25. A range limit for the constraint variable x_{con} can be added accordingly. In this research, a real-valued GA (Houck, 1996) is used instead of a binary GA, since the real-valued GA parameters' values evolved by genetic operations remain within their corresponding range limits. By this transformation, the optimization process can become more efficient if the design space reachable by x_{con} is smaller than the design space reachable by the original design variable x_1 (in the current implementation, this validation is manually determined and stored in the component repository, but remains open to automated deduction in further research). Eqs. 3 and 4 of Figure 4-25 show an example

of this transformation, by which the constraint variable (diamond 'con.1' of Figure 4-24 (a)) is encoded as a design variable ranged between 4 and 12 (replacing the original design variable 'd' ranged between 1 and 20 millimeters as in Figure 4-22), into a genotype that is partially shown in Figure 4-24 (c).

$$(1)\quad x_{con} = f(x_1, x_2, \ldots x_n) \le 0 \longrightarrow 4 < \frac{D}{d} = x_{con1} < 12 \qquad (3)$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$(2)\quad x_1 = f'(x_{con}, x_2, \ldots x_n) \longrightarrow d = \frac{D}{x_{con.1}} \qquad (4)$$

$$x_{con} \in [-\infty, 0] \qquad\qquad x_{con.1} \in [4, 12]$$

Figure 4-25: Design constraints transformation

4.7.3 Fitness Function and Design Results

Through automated modeling and model transformation, state space equations (Figure 4-26) are generated from the CD-Graph for the weighing machine design. As shown there are 7 integrative states and 6 derivative states. Two outputs were defined as functions of two states prior to the computational design efforts, y_1 (footpad displacement) and y_2 (angle of dial rotation). In terms of these two outputs there are three design targets as shown with the dashed lines in Figure 4-27 (a). With the input of 100 lb at the footpad they are: 1) Dial rotation equals to 1 radian: $f_1(y_1)$; 2) the footpad displacement is less than 2 cm (as close to 0 as possible): $f_2(y_2)$; 3) Dial settles down within 3 seconds: $f_3(y_1)$. The three cost functions are obtained from numerical calculations over a number of distributed sample points. The overall evaluation of a design is simply the weighted sum of these three cost functions. Note that the objective is

to “minimize” the overall cost to obtain the best design fitness. The green dotted line and the blue solid line (Figure 4-27 (a)) are the footpad and dial responses respectively of the best design. The evolutional optimization process is shown in Figure 4-27 (b) which indicates that the best design was found at the 57th generation. The total optimization process took about one hour and ten minutes with population size set as 50.

$$\dot{x}_1 = \frac{x_4}{sh1_c}$$

$$\dot{x}_2 = \frac{\left(\cos(x_5) le1_j le1_{la} \dot{z}_5 sh1_c sp1_c - \sin(x_5)^2 le1_{la}^2 x_2 rp1_r rp1_{tf}^2 sh1_c sp1_c\right.}{}$$

$$\cdots$$

$$\dot{x}_7 = \frac{(\sin(x_5) le1_{la} x_2)}{le1_j}$$

$$z_1 = \frac{(\sin(x_5) le1_{la} x_2 we1_i)}{le1_j}$$

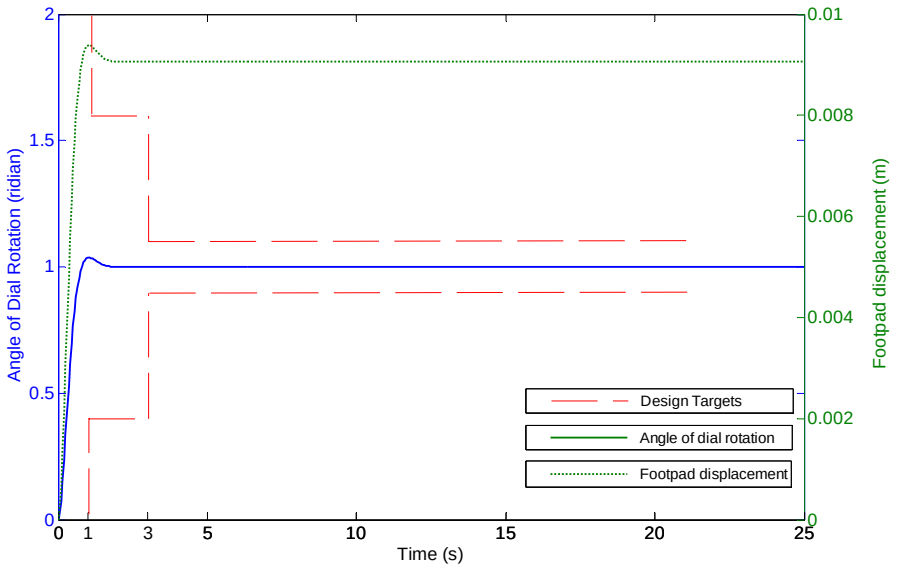
$$\cdots$$

$$z_6 = \frac{(\sin(x_5) le1_{la} le1_m x_2)}{le1_j}$$

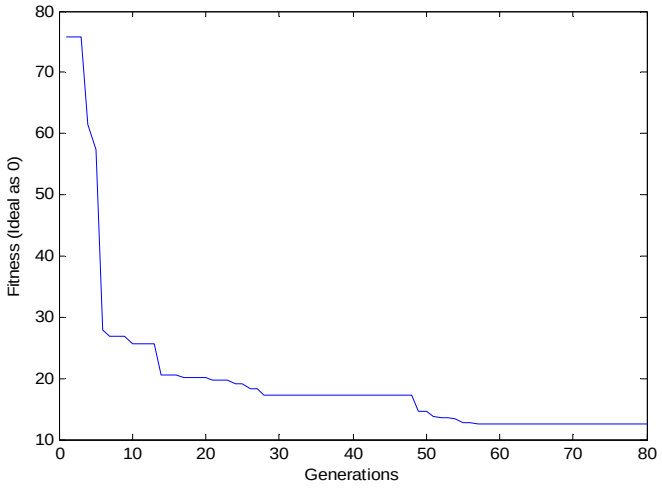
$$y_1 = x_6$$

$$y_2 = x_7$$

Figure 4-26: Equations generated from CD-Graph of the weighing machine by automated modeling based on modularized bond graph models. There are two outputs defined at the beginning of the design process, y1 (the footpad displacement) and y2 (the angle of the dial rotation).



(a): Response of the best design obtained by GA optimization. The green dot line and the blue solid line are the footpad and dial responses respectively.



(b): The evolutional optimization

Figure 4-27: Weighing machine design

Table 4-3: The parameters designed for the components Shaft and Spring

Comp.	Property	Value Designed	Constitutive Laws & Constraints	Value Assigned	Lower Limit	Upper Limit
Shaft	Shear Modulus: G (Pa)			7.31E+10		
	Density: ρ (kg/m3)			7.75E+03		
	Rot. Compliance: C (m/N)		= 1/(G*pi*D^4/(32*L)) = 2.7349e-003			
	Rot. Inertia: I (kg)		= ρ*pi*L*D^4/32 = 8.7770e-007			
	Length: L (m)	1.5038E-01			0.05	0.5
	Diameter: D (m)	9.3570E-03			0.002	0.02
Spring	Ratio Constraint: R	9.7535E+00	4<D/d<12		4	12
	Shear Modulus: G (Pa)			7.31E+10		
	Wire Diameter: d (m)		= D/R = 5.1881E-003		0.001	0.020
	Coil Diameter: D (m)	5.0602E-02			0.005	0.10
	Number of coils: N	9.6691E+00			4	20
	Compliance: C (m/N)		= (8*D^3*N)/(d^4*G) = 1.8923E-004			

The optimized results for details of the shaft and spring design are shown in Table 4-3. The values in the column "Value Assigned" are assigned to the corresponding properties (such as material density of the shaft) by designer's preference, while leaving other properties to be determined by the subsequent computational design efforts. Through the aforementioned strategies on preparing genotype, a proper set of properties were encoded into the genotype as design variables and the values of these design variables are shown in the column "Value Designed". The design constraints and constitutive laws are listed in the next column and used to either validate the values designed or calculate other intermediate properties. To reduce the design space, a range for each design variable was stored in the component repository based on common applications and open to designer's modification.

4.7.4 Experimental Setup

In the implementation of this research Matlab® is used as a programming platform and combined two existing public-domain software, MTT [Gawthrop and Bevan, 2003] and GAOT [Houck et al., 1995]. MTT is used to transform dynamics models from bond graph representation to another representation such as state space differential equations. GAOT is a genetic algorithm package that supports real-valued operations.

4.8 DISCUSSION

This paper introduces research leading to a computer-aided design tool in which engineering designers can conveniently test design concepts (topologies). A design representation called Conceptual Dynamics Graph (CD Graph) is used as the interface that relates a topology of components to its system dynamics model. Based on CD-Graph, qualitative evaluation of computational generated design concepts needs to be addressed to screen out the valid design concepts for detailed parameter design, which is currently among our ongoing research.

To automatically model a design configuration, a generic model is predefined for each component that accommodates various interactions with the environment of this component. Virtual couplers of CD Graphs record the interaction format in which generic models of components are assembled topologically. Transformation between global and local coordinates is also captured in the generic models. However, further work is still needed to enrich our bond graph generic models to make this work more powerful. Currently generic models (of different complexities) for 10 components are implemented

in the system which includes Gear, Wheel, Motor, Lever, Spring, Pulley, Shaft, Bar, Rack and Generator.

Note that some commercial software packages like ADAMS and Dymola support automated modeling. However, they are incapable of modeling designs represented in a highly conceptual fashion like CD-Graph, which facilitates the simplification of the design configuration by lumping together the components with same functionality, and the choice of models with appropriate degrees of freedom. Also, there is limited flexibility in obtaining analytical models in a desired format (as is shown in the equations of Figure 4-26) to integrate with optimization that requires specific evaluation routines. Further, our bond graph approach provides more physical insights than these dynamics simulators due to the models' graph-based nature.

Automated optimization practice is built on pre-defined performance target(s). Some components may be specified in the to-be-finished topology (incomplete design) to carry the overall system inputs and outputs, for example, the footpad and dial in the weighing machine design.

4.9 CONCLUSIONS

In this chapter, a conceptual design instantiation procedure composed of automated modeling and computer aided optimization has been elaborated, which is leading to a computer-aided design tool in which engineering designers can conveniently test various design concepts to best meet the dynamics specifications of design problems. The input to the system is a graph of components (CD-Graph) where the components'

design variables are to be determined by the subsequent optimization based on the analytical system model generated automatically.

Automated bond graph modeling has been introduced that relieves the burden of understanding of the system dynamics of design configurations by leveraging generic models stored in a component repository. This chapter also discussed a systematic approach to automatically prepare a design genotype for the application of GA optimization using the stored design heuristics. This preparation can encode and decode proper design variables into and out the genotype in a sense that accounts for the existing design constraints and physical constitutive laws. An example of weighing machine design was used to elaborate the approach.

Chapter 5 MODEL COMPLEXITY

In a system design process, it is important to predict the essential behavior of the proposed system and compare the predicted performance with the performance specifications. This, in turn, requires an appropriate model, which should not be so oversimplified that it gives inferior if not completely inaccurate results, nor should it be so complicated that it provides little insight into the relations between the design parameters and system performance. Thus, one needs a model of suitable complexity so as to provide sufficient information about the essential behavior of the system. However, the selection of a proper model is not trivial. Typically, experienced analysts are required to develop the necessary models. Design engineers who do not have the required skills and/or experience and who are under time pressure would likely use modeling tools as part of the design process if such modeling tools are available.

5.1 MODEL COMPLEXITY CLASSIFICATION

Dynamics models are an abstract representation of the behavior of a system as it interacts with its environment. As demonstrated in the previous chapter, a component can have models of different complexities in terms of degrees of freedom according to the environments this component interacts with. Generic models by their nature lead to complex system models, which is an unwelcome effect of enabling generic automated modeling. In the conceptual design stage, there are no parameters available yet; however,

qualitative strategies can be used to select the models of proper complexities for system modeling.

The level of generic component model to be used in the modeling of a design configuration, which could be 3D (6 domains), 2D (3 domains) or even 1D (1 domain), simply depends on the design‘s potential range of motion. For example, 3D model needs to be used if the component is designed to move in a 3D path (e.g., a wheel on a sports utility vehicle, or the Euler disk problem); but 2D model can be used for simplicity if it is to move in a 2D plane (e.g., a planet gear, or a train wheel that travels along a rather straight line); or the model can be further simplified to 1D for some cases (e.g., a fly wheel used to store kinetic energy).

The range of motion can be qualitatively determined from the design’s CD-Graph. If the coupling types at all those domains that lead to a higher dimension are rigidly coupled with ground, then a lower dimension component model can be used. The activity of the non-mechanical domains can also be determined from the CD-Graph simply by checking the coupling types of these domains with its environment, which tells how complex a model needs to be in terms of degrees of freedom or number of domains. Further granularity may be pursued to have specific models for more specific application situations. For example, a shift gear used in the automobile transmission has two active domains including one translational domain and one rotational domain along the same axis.

Reducing model complexity in term of degrees of freedom is depicted in the plane “Parameters Unknown” of Figure 5-1. However, it is also important to note another type of complexity, which is the number of modes modeled for a component (in the plane

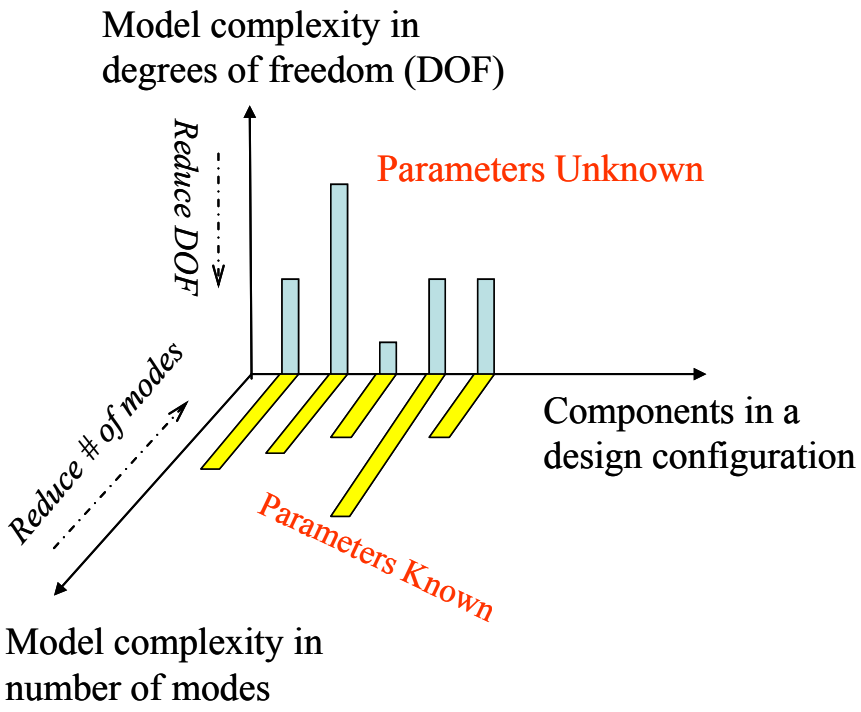


Figure 5-1: Modeling space of design configurations. A component can have models of multiple complexities in terms of number of modes or degrees of freedom.

"Parameters known" of Figure 5-1). The mathematical model of a system should predict the essential behavior of the real system relevant to the control problem. For example, for a DC motor position control problem, one can afford a low order model if the required closed-loop bandwidth of the system is low enough since the modes at high frequencies will not be excited. However, if high performance and fast response are required, a higher order model with more modes would be necessary in order to accurately predict the high frequency dynamics of the system. Thus, the mathematical model should have sufficient complexity: it should neither be oversimplified, nor too complex. This requires a means to determine the model order commensurate with the design goals. With given inputs and design parameters, some modes are dominant for the component or system's performance

with the effect that some other modes are negligible. The sequence of modes priority can vary significantly when the context (inputs and design parameters) changes.

5.2 MODEL ORDER DEDUCTION

5.2.1 General Approaches

There are basically two approaches to construct an appropriate model of a system. One approach to modeling is to begin with a high-order model of a physical system and apply mathematical transformations to pare the high frequency modes from the model. Such a model order reduction approach is quite powerful and commonplace; however, its use presupposes the existence of a high-order model to start with. Several model order reduction schemes exist and some are available in commercial packages such as Matlab. In contrast to this approach, model order deduction begins with a very low-order model of a system and adds state variables to the model - through the inclusion of inertial, compliant, and dissipative elements - until the performance predicted by the model within the frequency range of interest no longer changes (appreciably) as more state variables are added to the model.

Several researchers have addressed the problem of synthesizing models of appropriate complexity (order) with the premise that modeling tools could be used to systematically build proper models of systems consisting of interconnected components. Wilson and Stein [1992, 1995] discuss a model order deduction algorithm (MODA) to automatically build proper models of linear systems containing components represented by bounded or unbounded models. Given a frequency range of interest (FROI), MODA

performs a heuristic search to find an optimal system model such that the spectral radius is minimum out of all possible system models of equivalent complexity.

Ferris et al. [1994] developed EXTENDED-MODA which extends the concepts addressed by MODA by adding an accuracy criterion, in terms of the convergence of critical system eigenvalues, to the model synthesis algorithm and incorporating the modeling of hybrid systems. Wilson and Taylor [1995] addressed the issues in modeling nonlinear systems where they used describing-function methods. Wilson et al. [1995] developed frequency-domain model order deduction algorithm (FD-MODA) as an improvement over the selection of proper models of linear systems. FD-MODA uses the convergence of the frequency response of a system as a means of identifying its proper model.

As exemplified in the cited references, the research in model order deduction has the following context by assuming at the outset: *1) that the modeler is dealing with an electromechanical system that is assembled using a number of components (motors, gears, shafts); 2) and that a modeling technique exists that can be used to generate a system model once the components are specified.* The first assumption applies to the research activities not only at the "Parameters Known" plane but also the "Parameters Unknown" plane. The second assumption for research in the "Parameters known" is backed up by the research in the "Parameters Unknown" plane that aims for a modeling technique to generate a system model once the components are specified. In the literature of model order deduction, the test problems were intentionally restricted to one degree of freedom connection between any two components.

In fact, the technique based on the two assumptions can produce a number of dynamic models, which differ in the level of detail used to model each component. In the linear case, the “level of detail" of each component submodel translates into its order; e.g., a more detailed motor model may include inductive lag (one additional state variable), or a more detailed shaft model might include one or several modes (adding two state variables to its lumped-parameter model per mode). Component submodels are then combined to produce the overall system model. The problem addressed by model order deduction algorithms is deciding what minimal level of detail to use in each component submodel so that the resulting system model is suitably realistic.

Before this problem can be solved, one must specify what is meant by the term “suitably realistic." This consideration introduces the idea of a “model performance metric." Several of them have been proposed in the literature: MODA uses a FROI (frequent range of interest); EXTENDED-MODA uses a combination of a FROI and the convergence of critical system eigenvalues as model performance metrics; and FD-MODA makes use of the normalized change in the model's frequency response (sup norm of $\delta G(jw)$) as the criterion for judging the suitability of the system model as we iteratively increase the order of each component submodel and look for convergence.

5.2.2 State-Space Searching for Model Order Deduction

Similar to the design automation in [Campbell, 2000], the state-space searching concept also applies for modeling automation. Basically, given the physical configuration and component representation library, the three stages, generation and evaluation and

guiding, are iterated to find the proper model. Figure 5-2 is the sketch of the whole process of modeling automation to deduce a proper model using MODA algorithm.

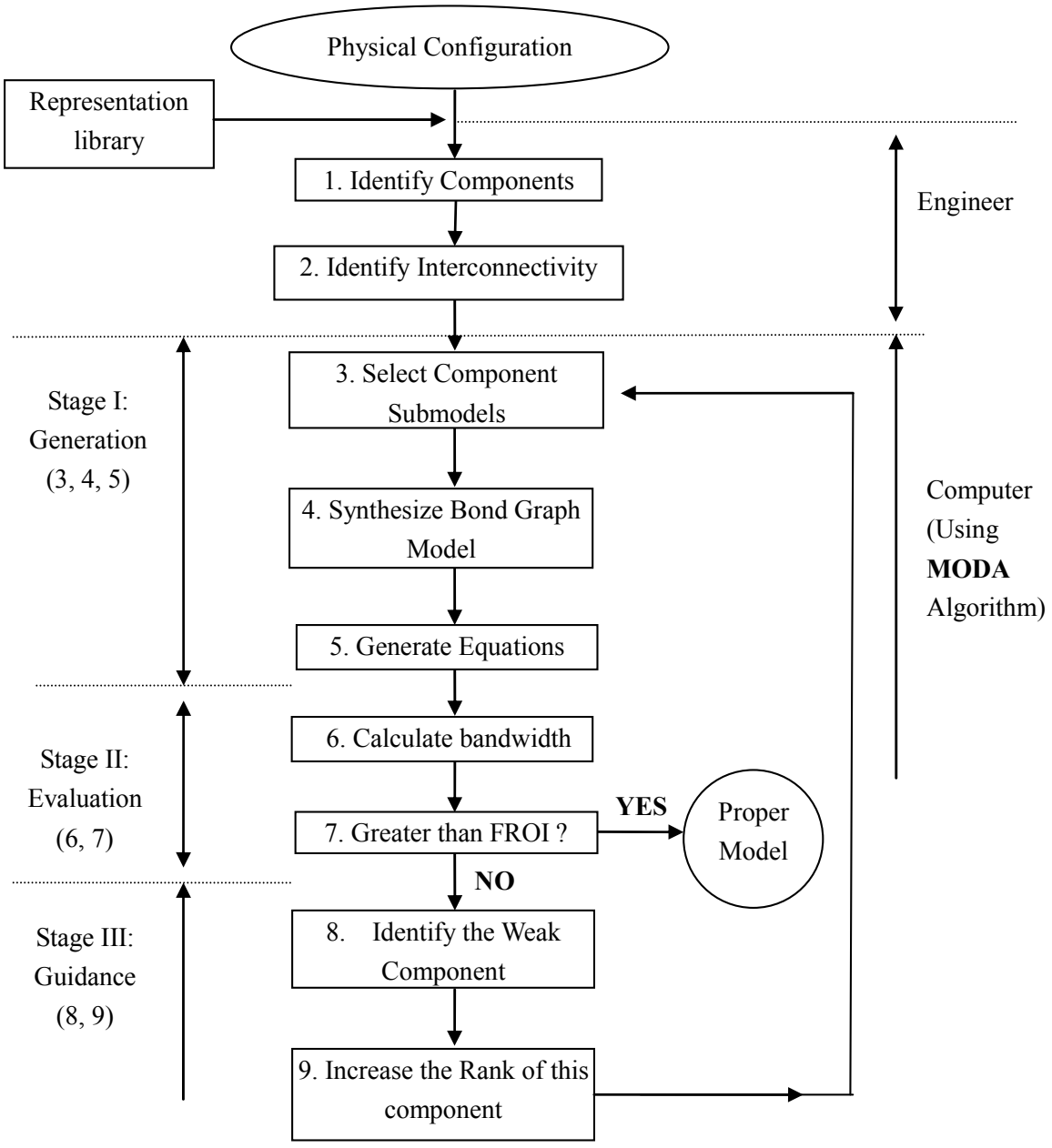


Figure 5-2: The scheme of modeling automation using MODA algorithm

The representation library shown consists of all the possible sub-models for any component of the design configuration. The task of engineers is to identify components of the design and their interconnectivity. The outcome of the identification can be represented as a graph with a node representing each component of the system. Each node can be replaced by the component model of variable complexity, and has a fixed number of ports to interact with other nodes and the system environment. Then the computing iteration begins from the stage I, generation, includes selecting the model (bond graph representation) for each component. These models are then synthesized to form the bond graph system model. Then, using the existing software like 20sim, or a self-developed package, the system model can be converted to equations. In the second stage, model evaluation, determine the system bandwidth to compare with the user-specified FROI. Finally following the guiding strategy, if not getting proper model yet, go ahead to identify the weak component and increase its rank to return to the first stage for another iteration.

5.2.2.2 Component Representation

An attribute variable called rank is associated with any component model to specify the complexity of this model. Larger ranks correspond to more complex sub-models of the component. The simplest model of a given component is its rank-0 model (a rigid body model). [Stein and Louca, 1996]

Components can be separated into two categories: bounded rank and unbounded rank. The bounded-rank component has only a finite number of independent energy storage elements, i.e. states. While the unbounded rank component may have rank from 0

to ∞. In a motor-pulley drive train example, the DC motor and the belt drive are considered as bounded-rank component, the torsional shaft is considered as unbounded-rank component.

The Belt-Drive consists of two sheaves and a massless, compliant belt. In the case when the compliance of the belt is included in the model there are three independent energy storage elements (rank-1). In the rank-0 case the belt is assumed be infinitely stiff. The two sheaves (Inertia 1 and 2) are now kinematically coupled into one independent energy storage element. Its bond graph is shown in Figure 5-3 with the shaded element removable to reach lower ranks.

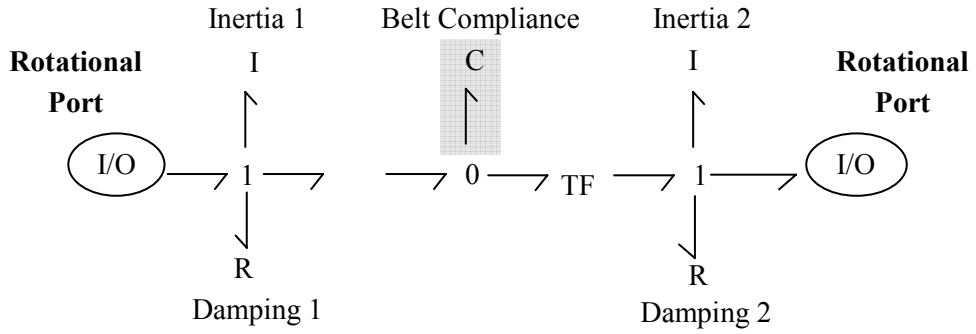


Figure 5-3: Rank 0-1 belt drive. [Stein and Louca, 1996]

The DC motor is modeled (Figure 5-4) with either one or two independent energy storage elements [Franklin and Powell, 1991]. When the motor is modeled with two independent energy storage elements (rank-1), the flux linkage is included in the model. In the rank-0 case the inductance of the winding is assumed negligible. Note that, in modeling the motor in this manner, effects such as heating of winding, saturation, cogging, etc. are not included in the model. They are implicitly assumed to be negligible

in the drive train application, but even higher order dynamics model may be developed to capture these effects.

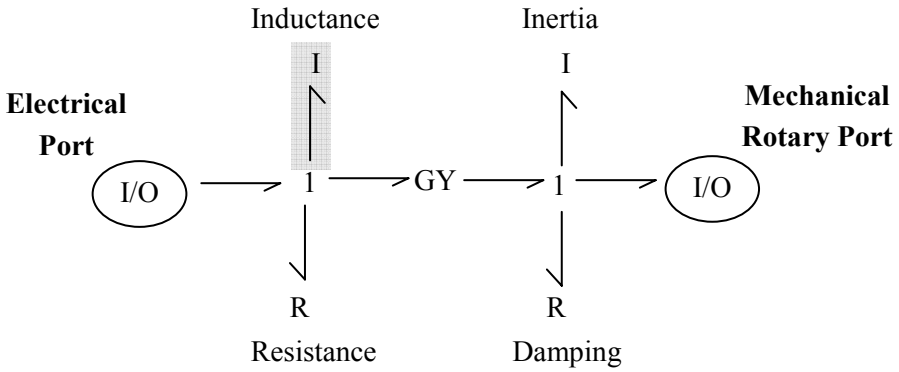


Figure 5-4: Rank 0–1 DC motor. [Stein and Louca, 1996]

The torsional shafts are unbounded rank components, which means the rank could vary from 0 to ∝. The magnitude of the individual inertias and compliances are obtained by the following equations [Rao, 1990]:

$$
\begin{aligned}
J_{shaft} &= \frac{\rho \times \pi \times L \times D^4}{32} \\
K_{shaft} &= \frac{G \times \pi \times D^4}{32 \times L} \\
J_i &= \frac{J_{shaft}}{N+1} \\
K_i &= N \times K_{shaft}
\end{aligned}
\qquad (5\text{-}1)
$$

where

- N = the rank associated with the shaft, N >= 0
- L = the length of the shaft
- D = the diameter of the shaft
- G = the shear modulus of the shaft
- J_i = inertia coefficients in the model

- K_i = spring rate coefficients in the model
- J_{shaft} = the torsional inertia of the shaft
- K_{shaft} = the torsional spring rate of the shaft

Note that for the torsional shaft, the rank N equals the number of torsional springs in the physical model. Its bond graph is shown in Figure 5-1.

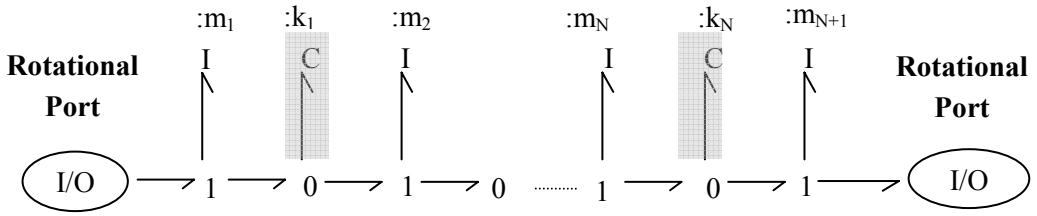


Figure 5-5: Rank 0–∝ shaft model. [Stein and Louca, 1996]

5.2.2.3 Model Generation

For the drive train system, where the interconnections are in series, bond graph formulation is handy in the sense that they can be combined systematically to build the overall system model. It is straightforward to generate the system model in bond graph form since the theory is that the bond graphs of all the components are interconnected based on the connections generated by the user during step 2. With this character, one can easily come up with models of systems whose subcomponent models are available for this purpose.

Given the system bond graph model and all the related parameter values (the initial state condition, element property, etc), it is not a hard problem now to use the existing software to generate the model equations and even perform dynamic simulations.

5.2.2.4 Model Evaluation

As showed in step 6, 7, 8 and 9 of Figure 5-2, for a candidate model, there must be a method to evaluate it and tell if this is the expected proper model. If the model meets the requirement, the searching process then comes to the end otherwise following the guiding strategy, it will go back to step 3 and start over again with increasing the rank of a specific component. The problem of modeling a complex system properly is actually transformed to a search algorithm given a model performance metric and models of different levels for each component.

MODA uses the relation between the FROI (frequency range of interest) and the spectral radius of eigenvalues as its performance metric and search for the combination of subsystem models such that the overall system has all its eigenvalues within the FROI. The model thus synthesized is optimal in the sense that 1) the system model assembled from the submodels has the minimum spectral radius (ρ_{model}) out of all possible models of equivalent complexity, and 2) any increase in model order would result in a spectral radius beyond a specific FROI. [Wilson and Stein, 1992]

The spectral radius, ρ_{model}, is the Euclidian norm of the largest eigenvalue of a state matrix, and is effectively the highest complex-scalar input frequency to which a model can respond. Karnopp et al. [1990] used the FROI to determine the required complexity of a model synthesized from a configuration description. Assuming that we have reasonable knowledge of input frequency content, and then we shall know which modes will be excited significantly. As a rule of thumb, they suggest that modes be retained up to a frequency at least a factor of 2, but no more than a factor of 5, higher than the frequency of excitation. That is, the model should contain sufficient modal

information to predict how the actual system would respond to frequencies in a range between 0 and $5*\omega_{max\_excitation}$. Thus, $5*\omega_{max\_excitation}$ becomes the required model bandwidth (FROI). Using this criterion from Karnopp et al. [1990], if $\rho_{model} > FROI$, the model can accurately respond to frequencies up to $\omega_{max\_excitation}$.

So given a model, as the step 6 and 7 of Figure 5-2, the model bandwidth (ρ_{model}) will be evaluated to compare with the FROI, and to tell whether this model reaches the requirement.

5.2.2.5 The Guiding Strategy

The guiding strategy will be used to determine which component's rank should be increased to start a new iteration if the current model is not proper ($\rho_{model} < FROI$). Consider the problem of finding a Proper Model for an N-component configuration. An exhaustive search strategy for this combination of component ranks is likely to be inefficient, because it results in a potentially very large search space.

A heuristic to guide the search (steps 8 & 9 of Figure 5-2) can greatly reduce the search-space. The example search space for a three-component configuration that results from using the heuristic is shown in Figure 5-6. The heuristic is derived from the ρ_{model} of the model associated with each node. The search starts from the root, a rank-0 configuration, and creates a set of test models that correspond to individually increasing the rank of each component (if applicable) in the configuration (level 1), where level-1 corresponds to a configuration whose component ranks sum to 1. The idea is to select the model with the minimum ρ_{model} (at each level), and to use this model as a starting point

for finding the next model. The model with the minimum ρ_{model} is found by testing the effect on ρ_{model} of increasing the rank of each component. The component that causes the smallest increase in ρ_{model} when the rank of this component is increased is referred to as the weak-dynamic link component. The search concludes when the bandwidth of all models at a given level M is greater than the FROI, and the solution is the model (at level M-1) with the minimum bandwidth. [Wilson and Stein, 1992]

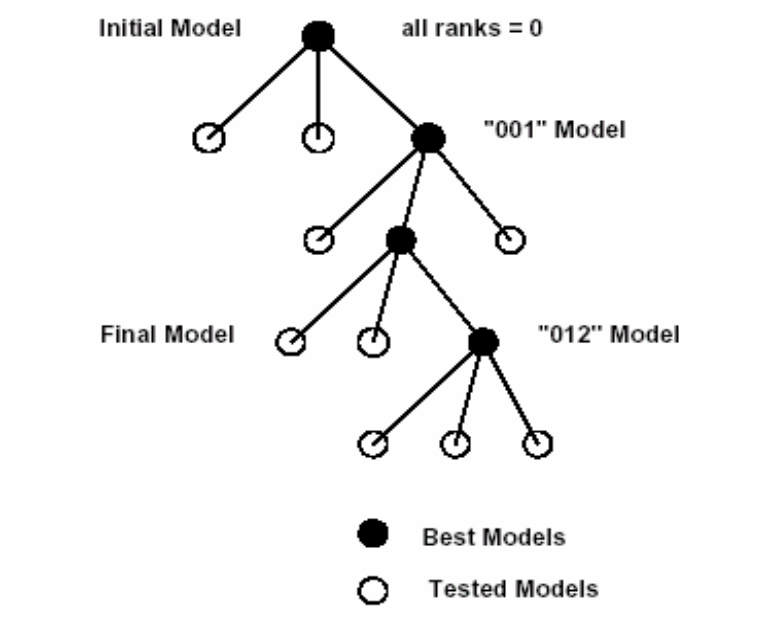


Figure 5-6: Reduced search space. [Wilson and Stein, 1992]

This search guiding technique fits in the category known as best first search. Using this technique the number of total trial solutions (models) needed to evaluate an N-component configuration (assume for simplicity that all components are of type unbounded-rank.) to a level of M equals:

$$N_{total\text{-}models} = 1 + N_{components} * (M_{level} + 1) \quad (5\text{-}2)$$

5.2.3 Limitation of MODA

Although MODA provide a way to derive Proper Model, there are some limitations:

1) The models synthesized by MODA have all the system eigenvalues below a given FROI. However, the accuracy of these eigenvalues is not guaranteed. As one increases the component model ranks, the eigenvalues generally converge to those of a high-order truth model.
2) No analytical method to prove that the Proper Model generated from MODA has the minimum spectral radius ρ_{model} for a given complexity level.
3) No analytical justification that it is the best to increase the rank of the component which leads to the minimum ρ_{model}, since the residue of a mode can also play a role.

The first limitation is overcome by Extended-MODA [Ferris et al., 1994] though keeping increasing the ranks of the components to make the eigenvalues below the FROI converge into some small value. For the second limitation, Wilson reported that no exception found in the trial examples in his Ph.D dissertation. All three limitations are overcome by another algorithm FD-MODA (Frequency Domain Method for Model Order Deduction Algorithm) [Wilson, 1995]. The heuristics used here is to increase the rank of the component which makes the largest frequency response change (Magnitude plot in Bode diagram) and keep increasing the sum of the ranks until the frequency response

converges into some specified tolerance TOL. The performance metrics of Extended-MODA and FD-MODA are described in Appendix D.

In this chapter, a qualitative strategy was introduced to select component models with proper complexity in terms of degrees of freedom; and an iterative searching algorithm (originally developed by other researchers) including model generation, evaluation and guiding provides a sound frame to implement the procedure for quantitatively selecting component models with proper complexity in terms of number of modes, where different performance metrics can be used to determine when to terminate the searching process.

Chapter 6 CONCLUSIONS

This dissertation has introduced the A-ODDS automated approach to the optimal design of dynamic systems. The creation of designs occurs as a result of evolvable design agents that are folded into a stochastic iterative process capable of adapting to changes in user preference. The innovations of A-ODDS are based on a number of related research topics including Probabilistic Approaches, Multi-Agent Systems, Bond Graphs, Genetic Algorithms, Physical system modeling, Stochastic Optimization, etc.. Table 6-1 provides a summary of these related research topics and the A-ODDS subsystems they influence. The table also shows how A-ODDS expands upon or diverges from these topics to create the unique constituents of the theory. The goal of A-ODDS is to combine open-ended topology generation, automated modeling, and integrated optimization (iteratively guided searching) for Automated Optimal Design of Dynamic Systems.

6.1 TOPOLOGY GENERATION

A-ODDS research proposes an evolutionary design method that combines a probability based decision strategy and design grammars (production rules) into a "design agent" for system development. The decision strategy can be evolved by applying a genetic algorithm (GA) to facilitate the exploration of a multi-domain design space in a topologically open-ended manner, and still efficiently find appropriate design configurations. This method is applied to make dynamic system designs using bond

Table 6-1: Derivation of the A-ODDS theory and implementation.

A-ODDS Subsystems	**Design generation**	**Automated Modeling**	**Searching Guidance**
Related Work	Multi-agent Systems, Genetic Algorithms, Genetic Programming, Grammar Rules, Probabilistic Approaches (PBIL, cGA, BOA, etc.)	Automated Modeling (MODA, EX-MODA, FD-MODA), Bond Graphs, Multi-attribute Utility Theory,	Tabu Search, Stochastic Optimization, Reinforcement Learning
A-ODDS Research	Evolvable Agents are goal-directed and work within a framework where grammar rules assist in generating diverse design solutions.	Generic component models (bond graphs) are created to facilitate automated modeling of design configurations.	GA and Statistical method are used to guide "Black box" design searching.

graphs. Experimental results in designing RC low pass filters show that design strategies demonstrate steady performance in terms of the overall fitness of its top designs. A good design strategy/agent has better chance of producing superior designs.

Unlike genetic programming where topology generation and parameter tuning happens at the same time, in A-ODDS they are separated as two consecutive stages, which allows better usage of analytical models obtained from the topology design and flexible choice of the optimization method for parameter tuning. This tuning method can be, generically speaking, stochastic or deterministic or a hybrid of both for better performance in terms of design quality and speed. Experimental results by RCL low pass filter design show that with gradient based optimization this two-stage approach can be significantly faster than regular genetic approaches on reaching designs of the same quality. With regard to the empirical results, it should be noted that it is incorrect to say

the approach introduced in this research will always outperform a traditional GA or GP that directly encodes designs. Genetic Programming has good performance especially when the design space becomes highly rugged, non-continuous and gradient based nonlinear optimization can not work well.

In A-ODDS, the design agents are encoded and evolved instead of designs, which facilitates a learning process for the design agents on how to efficiently generate designs to meet the user requirements. This evolvable agent based design approach is much simpler than GP in terms of comprehension and implementation for open-ended system development since there will be no genetic tree operations, no strict type matching and less pre-mature convergence.

As extensions of this research, I believe this approach can be further improved by leveraging a knowledge base for design agents through a rich and structured representation of design heuristics; and by expanding collaborative efforts between agents from agent evolution to the concrete process of creating designs. Further, it is also interesting to explore other learning algorithms instead of GA to evolve a design agent in the future work of this research.

6.2 AUTOMATED MODELING

In this research, the aforementioned agent-based design method was used for bond graph design, but not directly for electro-mechanical system design due to the lacking of a fast and robust design evaluation platform for electro-mechanical designs. This evaluation platform is also strongly desired by design engineers in order to automatically and efficiently evaluate various design concepts manually generated at the

conceptual design stage. An important task in dynamic system design is creating a repository of electro-mechanical components that stores the knowledge (grammar rules) of how interactions between components happen [Bohm & Stone, 2003, 2004] and how to model the components and the interconnections using bond graphs within various coupling contexts [Wu et al. 2007]. This repository will provide a broader application for the proposed agent-based design method.

In the A-ODDS research, a conceptual design instantiation procedure composed of automated modeling and computer aided optimization was elaborated, which is leading to a computer-aided design tool in which engineering designers can conveniently test various design concepts (topologies) to best meet the dynamics specifications of design problems. The input to the system is a graph of components (CD-Graph) where the components' design variables are to be determined by the subsequent optimization based on the analytical system model generated automatically.

A CD-Graph is composed of components and virtual couplers, where the components are associated with detailed bond graph models of multiple complexities, and the virtual couplers have the information on how two components interact with each other at each defined domain. The CD-Graph provides a platform for conceptual design synthesis and its further dynamics performance analysis. With a graphical user interface, designers could drag and drop components from the component repository, and specify how the components are to be connected. The detailed coupling type at each domain could be decided intelligently by the automatic detection of connection compatibility of components, which can be further clarified with the designer's instructions if necessary. Future research is needed to derive from CD-Graphs the qualitative information such as

degrees of freedom (applying Gruebler's equation), number of system states, energy flow patterns and possible operation modes to select the viable design concepts for detailed parameter design,

To automatically model a design configuration, a generic model is predefined for each component that accommodates various interactions with the environment of this component. Transformation between global and local coordinates is also captured in the generic models. Currently in the component repository generic models (of different complexities) for 10 components are implemented that include Gear, Wheel, Motor, Lever, Spring, Pulley, Shaft, Bar, Rack and Generator. In this research, automated bond graph modeling of design configurations helps relieve the burden of understanding of the system dynamics by leveraging component generic models. Further work is still needed to enrich the component repository to make this work more powerful.

Based on the resulting dynamics model from the automated modeling process of a CD-Graph design, the following step is to automatically invoke an optimization process to determine the choices for the design variables in each of the components pulled from the repository into the configuration. The symbolic equations can be derived automatically from bond graphs and used as the evaluation function in such an optimization where the initial specifications would represent the objective function values to be optimized. This research also discussed a systematic approach to automatically prepare a design genotype for the application of GA optimization using the stored design heuristics. This preparation can encode and decode proper design variables into the genotype in a way that accounts for the existing design constraints and physical constitutive laws.

Automated optimization practice is built on pre-defined performance target(s). Some components may be specified in the to-be-finished topology (incomplete design) to carry the overall system inputs and outputs, for example, the footpad and dial in the weighing machine design. The computational power of combining automated modeling and optimization will thus eliminate the need to divide the design efforts prematurely and lead to more integrated and robust designs with less time and design effort.

6.3 SEARCHING GUIDANCE

In many situations searching for designs is a "black box" problem, which means no information about the pattern of the design space is known except the inputs to the design (black box) and the expected outputs. In this research, genetic inspired approaches are used for guiding the design searching process. In Chapter 3, the design agents store probabilities on what decisions to make at a certain design stage (choosing applicable operators, operands, and operating locations). These probabilities can be updated (evolved) through GA operations using the fitness of the designs generated by a design agent as the fitness of this design agent. This evolving process allows a more directed and efficient search for successful design alternatives. Further, in Chapters 3 and 4, a genetic algorithm is used for parameter tuning when gradient based optimization is not applicable. As demonstrated, the genetic approach works robustly for this problem domain.

The range of learning algorithms for "black box" design is large and intricate and lends itself to future research. However, if certain information about the pattern of the design space is known, a guidance strategy can implemented for an effective design

searching process. This type of design practice can be called “grey box” design. The simplest example is that a golden section searching method can be applied to find the extreme value of a quadratic function. Here the information we know about the solution space is that “this is a quadratic function”. This pattern-guided searching is more effective than exhaustive searching or other genetic approaches. [Rai, 2006] researched how to better understand a system through design of experiments, that aimed to turn the “black box” into a “grey box”.

6.4 LIMITATIONS

In the proposed automated design process, computational power is leveraged to effectively reach design solutions. However, this process is not “completely” automated since it still needs the instructions from human designers. These instructions include the application-specific design specifications and the strategies to be used to evaluate designs’ performance (i.e., how to express the design specifications as a function of the design inputs and system models), which cannot be captured in the component repository and the general design process. Further, designers can have the options to assert values to design parameters or leave them to be tuned by the proposed process through automatic genotype encoding and decoding.

Although an Euler disk problem has been simulated in this research, more complex systems like 3D robot design with multiple degrees of freedom (such as robot arm) currently have not been investigated in detail, which usually have multiple components connected together to generate 3D motions at the end-effectors. Also, The design tool proposed in this research aims at rigid body dynamic system design and the

component models are all developed based on this assumption. However, very often in mechanical designs, in addition to rigid body dynamics, other design requirements need to be considered, which include component fatigue, strength, manufacturability, or others in the areas of fluid dynamics, electromagnetism, optics, etc. In order to automate a design process that involves domains other than rigid body dynamics, a solver in that domain is needed to automatically evaluate design candidates. Engineers use FEM (Finite Element Methods) for strength analysis and CFD (Computational Fluid dynamics) for fluid system dynamics analysis. It is desirable in future research to integrate these computational methods into this automated design framework to allow powerful design evaluation through simulations in different domains.

6.5 CONTRIBUTIONS

This dissertation has presented the challenges for automated optimal design of dynamic systems. The innovations of A-ODDS set forth a new frontline towards this goal and help understand the possibility of automating dynamic system design.

The **contributions** of this doctoral research are:

1. An evolvable agent based approach deployed to work within a framework where grammar rules assist in generating diverse design solutions.
2. A design representation CD-Graph created to represent design configurations to facilitate the automatic design evaluation.
3. Generic component models (bond graphs) created to allow automated modeling of design configurations.

4. An approach to automatically prepare genotypes to provide convenience for applying GA optimization to a design problem.
5. (2+3+4) A method to automatically evaluate and instantiate design concepts of dynamic systems by combining automated modeling and automated optimization.
6. (1+5) A method for automated optimal design of dynamic systems composed of topology generation and topology evaluation.

APPENDIXES

A: MTT: AN OPEN-SOURCE BOND GRAPH TOOL

In the context of software, it has been said that one good tool is worth many packages. UNIX is a good example of this philosophy: the user can put together applications from a range of ready made tools. Licensed under the GNU General Public License [Free Software Foundation, 1991], MTT is a set of Model Transformation Tools [Gawthrop and Smith, 1996] under UNIX environment, each of which implements a single transformation between system representations.

After gaining an understanding of the system that the modeler wishes to represent, it is generally possible to create a suitable bond graph using a pencil and paper. The task for the modeler is then to translate the design into a form that can be implemented on a computer. Unfortunately, translation of conceptual models into computer-readable formats is an area where many software tools disappoint their users, requiring them to work within constraints imposed by the software developers. If the software designers do not use the same concepts or notations as the modeler, or if they have not included advanced features, such as vector bonds, bi-causality or bond graph inversion, the user generally has no choice but to request that the software vendor alter the software to behave as they wish. If the vendor is unwilling to do this, the user must either work within the limitations of the software, or seek alternative tools. With open source

software, however, the user has another option. When the source code is available to be viewed and modified, the user is able to study the code and adapt it to meet the user's exact requirements. Thus, after creating a representation of a system using a pencil and paper, the user is free to adapt the toolset to understand their conceptual model to match the software designer's interpretation [Balance et al. 2005].

Introduction

Transformations in MTT are accomplished using appropriate software (e.g. Octave/Matlab, Reduce) encapsulated in UNIX Bourne shell scripts. The relationships between the tools are encoded in a Make File; thus the user can specify a final representation and all the necessary intermediate transformations are automatically generated.

The software tools support a range of techniques including modeling, simulation, control system design and system identification. Particular care has been taken to handle large systems using a hierarchical approach.

Figure A.1 gives the paradigm with which MTT is developed [Gawthrop and Smith, 1996]. In particular, modeling is viewed as a sequence of transformations between representations of physical systems. The sequence of transformations starts with the physical system itself and ends with the desired model. The purpose of MTT is to provide support for generating one representation (in a relevant language) from another whilst leaving Transformation1 to the user. Representation1 is a hierarchical bond graph model of the system; the subsequent representations may include those listed in Figure A.2. Each representation can itself appear in a number of different (computer) languages

including those in Figure A.3. MTT provides translation tools for the conversion of a representation in one language to a different language where appropriate. The final model depends on its purpose, which could be simulation, control design, process design, analysis, etc. In this dissertation research, performance analysis is the main goal of dynamic system modeling to evaluate a conceptual design configuration.

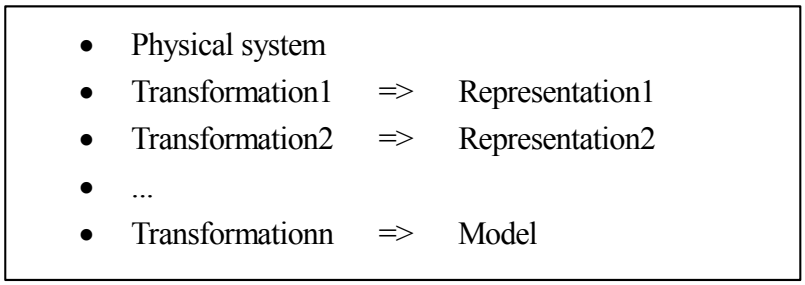


Figure A.1: Modeling as a sequence of transformations

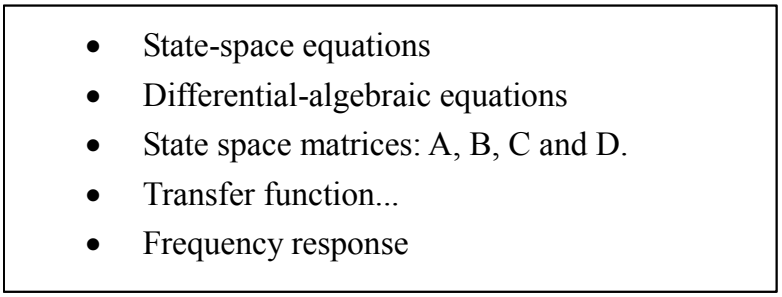


Figure A.2: Different representations of the same system

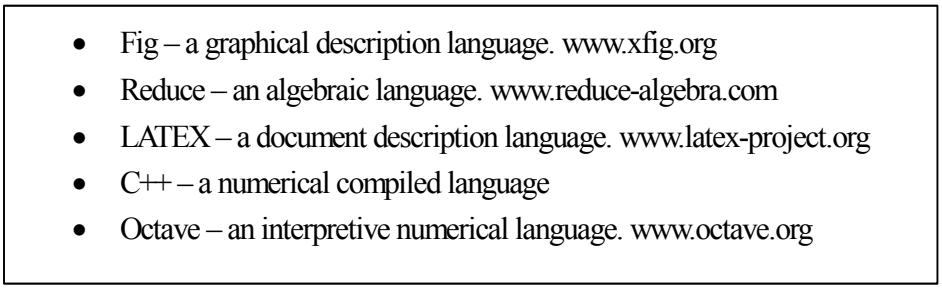
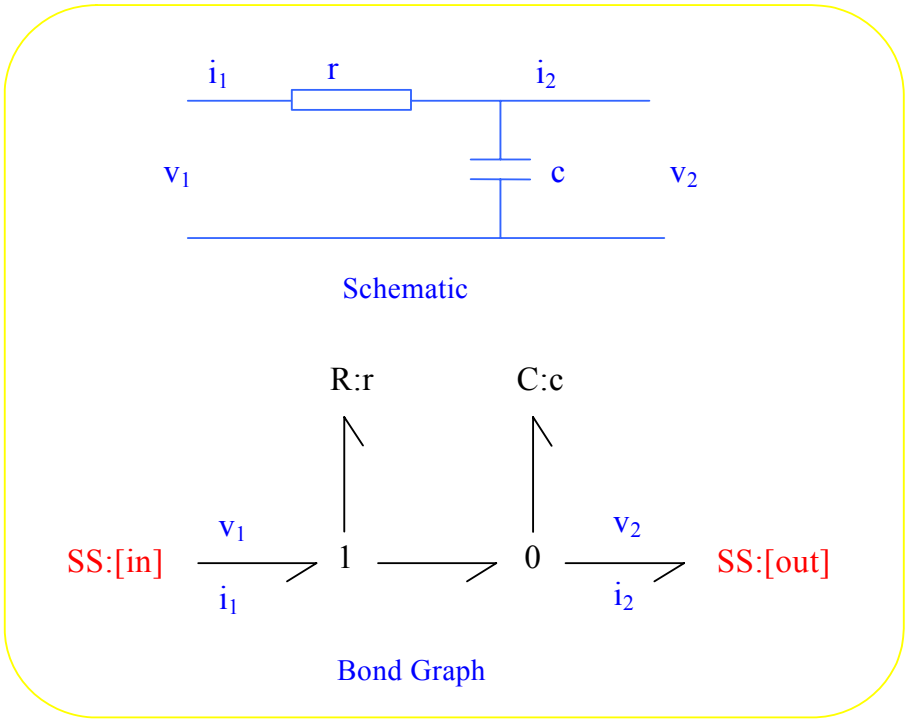


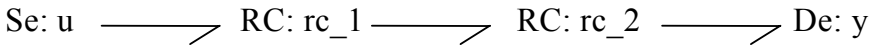
Figure A.3: Different languages describing the same representation

Example

As an introductory example [Balance et al. 2005], Figure A.4 shows two RC circuits modeled in a hierarchical fashion. Figure A.4 (a) shows a single RC circuit with two ports labeled SS:[in] and SS:[out] and Figure A.4 (b) shows the overall system driven by a single voltage source Se:u and the output voltage is measured by the ideal detector element De:y.



(a) Subsystem bond graph



(b) System bond graph

Figure A.4: Dual RC circuit [Balance et al. 2005]

The generic MTT command is:

mtt system representation language

Thus, for example:

mtt rc2 sm view

gives the state-space representation

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Where:

$$A = \begin{pmatrix} \dfrac{(-(r_1 + r_2))}{(c_1 r_1 r_2)} & \dfrac{1}{(c_2 r_2)} \\ \dfrac{1}{(c_1 r_2)} & \dfrac{(-1)}{(c_2 r_2)} \end{pmatrix}$$

$$B = \begin{pmatrix} \dfrac{1}{r_1} \\ 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & \dfrac{1}{c_2} \end{pmatrix}$$

$$D = (0)$$

and

mtt rc2 tf view

gives

$$G = \left(\frac{1}{(c_1 c_2 r_1 r_2 s^2 + c_1 r_1 s + c_2 r_1 s + c_2 r_2 s + 1)} \right)$$

Further details about MTT commands and examples of their use can be found in the MTT manual [Gawthrop and Bevan, 2003].

B: GENETIC APPROACHES

Genetic algorithms

Genetic algorithms (GAs) are biologically motivated adaptive systems which are based upon the principles of natural selection and genetic recombination. A GA combines the principles of survival of the fittest with a randomized information exchange. It has the ability to recognize trends toward optimal solutions, and to exploit such information by guiding the search toward them. In the standard GA, candidate solutions are encoded as fixed length vectors. The initial group of potential solutions is chosen randomly. These candidate solutions are allowed to evolve over a number of generations. At each generation, the fitness of each solution is calculated; this is a measure of how well the solution optimizes the objective function. The subsequent generation is created through a process of selection, recombination, and mutation. The solutions are probabilistically selected for recombination based upon their fitness. General recombination (crossover) operators merge the information contained within pairs of selected "parents" by placing random subsets of the information from both parents into their respective positions in a member of the subsequent generation. Although the solutions with high fitness values have a higher probability of selection for recombination than those with low fitness values, they are not guaranteed to appear in the next generation. Due to the random factors involved in producing "children" solutions, the children may, or may not, have higher fitness values than their parents. Nevertheless, because of the selective pressure applied through a number of generations, the overall trend is towards higher fitness solutions. Mutations are used to help preserve diversity in the population. Mutations

introduce random changes into the solutions. A good overview of GAs can be found in [Goldberg, 1989] [De Jong, 1975].

Population-based Incremental Learning

Population-Based Incremental Learning [Baluja, 1994] is a technique derived by combining aspects of genetic algorithms and competitive learning. This algorithm and its variants have been shown by Baluja to significantly outperform standard GA approaches on a variety of stationary optimization problems ranging from toy problems designed specifically for GA's to NP-complete problems [Baluja, 1997]. It has also been applied to various real-world applications including autonomous highway vehicle navigation. PBIL's most significant difference from standard genetic algorithms is the removal of the *population* found in GA's. A GA's population can be thought of as implicitly representing a probability distribution of alleles over genes. PBIL takes this implicit distribution and makes it explicit by dispensing with the population and instead maintaining a set of probabilities for alleles over genes. In the common case of binary genes (two alleles per gene, **0** and **1**) this set is simply a vector containing the probability for each gene that the allele is a **1**.

Learning in PBIL

Learning in PBIL consists of using the current probability distribution to create N individuals. These individuals are evaluated according to an objective function. The "best" individual is used to update the probability vector, increasing the probability of producing solutions similar to the current best individual. The update rule is essentially the same as that used in Learning Vector Quantization (LVQ) [Kohonen, 1989]. This

process of generation, evaluation, and update is repeated until some stopping criterion is met. Details of the algorithm and possible variations can be found in Baluja [1994].

PBIL and GA on Convergence

One key feature of the *early* generations of genetic optimization is the parallelism in the search; many diverse points are represented in the population of points during the early generations. When the population is diverse, crossover can be an effective means of search, since it provides a method to explore novel solutions by combining different members of the population. Because PBIL uses a single probability vector, it may seem to have less expressive power than a GA using a full population, since a GA can *represent* a large number of points simultaneously. A traditional single population GA, however, would not be able to *maintain* a large number of points. Because of sampling errors, the population will converge around a single point. This phenomenon is summarized below:

Diversity in the population is crucial for GAs. By maintaining a population of solutions, the GA is able—in theory at least—to maintain samples in many different regions. Crossover is used to merge these different solutions. A necessary (although not sufficient) condition for crossover to work well is diversity in the population. When diversity is lost, crossover begins to behave like a mutation operator that is sensitive to the convergence of the value of each bit [Eshelman, 1991]. If all individuals in the population converge at some bit position, crossover leaves those bits unaltered. At bit positions where individuals have not converged, crossover will effectively mutate values in those positions. Therefore, crossover creates new individuals that differ from the

individuals it combines only at the bit positions where the mated individuals disagree. This is analogous to PBIL which creates new trials that differ mainly in positions where prior good performers have disagreed.

Methods to Preserving Diversity

One method of avoiding convergence is to use a parallel GA (pGA). Many studies have found pGAs to be very effective for preserving diversity for function optimization [Cohoon et *al.*, 1988][Whitley et *al.*, 1990]. In the pGA, a collection of independent GAs, each maintaining separate populations, communicate with each other via infrequent inter-population (as opposed to intra-population) matings. pGAs suffer less from premature convergence than single population GAs. Although the individual populations typically converge, different populations converge to different solutions, thus preserving diversity across the populations. Inter-population mating permits crossover to combine solutions found in different regions of the search space.

Parallel PBIL (pPBIL) [Baluja, 1996] yield similar performance improvements to those achieved in pGAs. Multiple PBIL evolutions are simulated by using multiple probability vectors to generate solutions. To keep the evolutions independent, each probability vector is only updated with solutions which are generated by sampling it.

PBIL is an approach that explicitly preserving diversity while GA is an in-explicit approach that maintaining large populations. However, it is incorrect to say that PBIL will always outperform a standard GA. The main advantage of PBIL as claimed in [Baluja 1996] is that PBIL is much simpler than GAs.

Genetic Programming

Genetic programming is an extension of the genetic algorithm. In genetic programming, the genetic algorithm operates on a population of computer programs of varying sizes and shapes [Koza, 1992; Koza and Rice, 1992].

Genetic programming is an effective way to generate design candidates in an open-ended, but statistically structured manner. A critical aspect of the procedure is a fitness measure, which must guide the evolution of candidate designs toward a suitable result in a reasonable time. There have been a number of research efforts aimed at exploring the combination of genetic programming with physical modeling to find good engineering designs. Perhaps most notable is the work reported in [Koza et al. 1997]. He presents a single uniform approach using genetic programming for the automatic synthesis of both the topology and sizing of a suite of various prototypical analog circuits, including low-pass filters, operational amplifiers and controllers. This system has already shown itself to be extremely promising, having produced a number of patentable designs of useful artifacts.

The design process for analog circuits begins with a high level description of the circuit's desired behavior and characteristics and entails creation of both the topology and the sizing of a satisfactory circuit. The topology comprises the gross number of components in the circuit, the type of each component (e.g., a resistor), and a list of all connections between the components. The sizing involves specifying the values (typically numerical) of each of the circuit's components.

Starting with thousands of randomly created computer programs that represent designs, genetic programming progressively breeds a population of designs over a series

of generations. Genetic programming applies the Darwinian principle of survival of the fittest, analogs of naturally occurring operations such as sexual recombination (crossover), mutation, gene duplication, and gene deletion, and certain mechanisms of developmental biology. The computer programs are compositions of functions (e.g., arithmetic operations, conditional operators, problem-specific functions) and terminals (e.g., external inputs, constants, zero-argument functions). The programs may be thought of as trees whose points are labeled with the functions and whose leaves are labeled with the terminals. Genetic programming breeds computer programs to solve problems by executing the following three steps: [Koza et al., 1997]

(1) Randomly create an initial population of individual computer programs.

(2) Iteratively perform the following substeps (called a *generation*) on the population of programs until the termination criterion has been satisfied:

(a) Assign a fitness value to each individual program in the population using the fitness measure.

(b) Create a new population of individual programs by applying the following three genetic operations. The genetic operations are applied to one or two individuals in the population selected with a probability based on fitness (with reselection allowed).

(i) Reproduce an existing individual by copying it into the new population.

(ii) Create two new individual programs from two existing parental individuals by genetically recombining subtrees from each program using

the crossover operation at randomly chosen crossover points in the parental individuals.

(iii) Create a new individual from an existing parental individual by randomly mutating one randomly chosen subtree of the parental individual.

(3) Designate the individual computer program that is identified by the method of result designation (e.g., the *best-so-far* individual) as the result of the run of genetic programming. This result may represent a solution (or an approximate solution) to the problem.

Genetic programming starts with an initial population (generation 0) of randomly generated computer programs composed of the given primitive functions and terminals. Typically, the size of each program is limited, for practical reasons, to a certain maximum number of points (i.e. total number of functions and terminals) or a maximum depth (of the program tree). The creation of this initial random population is, in effect, a blind random parallel search of the search space of the problem represented as computer programs. The computer programs in generation 0 of a run of genetic programming will almost always have exceedingly poor fitness. Nonetheless, some individuals in the population will turn out to be somewhat more fit than others. These differences in performance are then exploited. The Darwinian principle of reproduction and survival of the fittest and the genetic operation of crossover are used to create a new offspring population of individual computer programs from the current population of programs.

The reproduction operation involves selecting a computer program from the current population of programs based on fitness (i.e., the better the fitness, the more

likely the individual is to be selected) and allowing it to survive by copying it into the new population.

The crossover operation creates new offspring computer programs from two parental programs selected based on fitness. The parental programs in genetic programming are typically of different sizes and shapes. The offspring programs are composed of subexpressions (subtrees, subprograms, subroutines, building blocks) from their parents. These offspring programs are typically of different sizes and shapes than their parents.

The mutation operation creates an offspring computer program from one parental program selected based on fitness. One mutation point is randomly and independently chosen and the subtree occurring at that point is deleted. Then, a new subtree is grown at that point using the same growth procedure as was originally used to create the initial random population.

After the genetic operations are performed on the current population, the population of offspring (i.e., the new generation) replaces the old population (i.e., the old generation). Each individual in the new population of programs is then measured for fitness, and the process is repeated over many generations.

The *hierarchical character* of the computer programs that are produced is an important feature of genetic programming. The dynamic variability of the computer programs that are developed along the way to a solution is also an important feature of genetic programming. It is often difficult and unnatural to try to specify or restrict the size and shape of the eventual solution in advance. Moreover, advance specification or restriction of the size and shape of the solution to a problem narrows the window by

which the system views the world and might well preclude finding the solution to the problem at all.

Automated programming requires some hierarchical mechanism to exploit, *by reuse* and *parameterization*, the regularities, symmetries, homogeneities, similarities, patterns, and modularities inherent in problem environments. Subroutines do this in ordinary computer programs. Automatically defined functions [Koza, 1994a&1994b] can be implemented within the context of genetic programming by establishing a constrained syntactic structure for the individual programs in the population. Each multi-part program in the population contains one (or more) function-defining branches and one (or more) main result-producing branches. The result-producing branch usually has the ability to call one or more of the automatically defined functions. A function-defining branch may have the ability to refer hierarchically to other already-defined automatically defined functions. Since each individual program in the population consists of function-defining branch(es) and result-producing branch(es), the initial random generation is created so that every individual program in the population has this particular constrained syntactic structure. Since a constrained syntactic structure is involved, crossover is performed so as to preserve this syntactic structure in all offspring.

C: EULER DISK SIMULATION

```
% EulorDisk.m
% For Eulor disk simulation, This file invoke another .m file: EulorDiskdot.m

% by: Zhaohong Wu
% 09/09/2006

% Define & share system parameters
global Jinv Fxyz FXYZ g R thick rho_s M I

% set the initial values .

Wx0 = pi/8;                          %roll angular velocity
Wy0 = 0;                             %tilt
Wz0 = 2*pi ;                         %pitch

psi0 = 0;                                          %roll angle
theta0 = 0; % 10*pi/180;          %tilt
phi0 = 0;                                          %pitch
alpha0 = [psi0; theta0; phi0];

FX = 0;
FY = 0;
FZ = - M*g;
FXYZ = [FX; FY; FZ];                               %Force applied

g = 9.81;                            % m/s^2              Acceleration of gravity
rho_s = 7830;                 % kg/m^3           Density of steel
R = 0.0735/2;                 % m                      Radius of disk
thick = 0.0127;               % m                      Thickness of disk
M = rho_s*pi*R^2*thick;               % kg                 mass of the disk

Jx = 0.5*M*R^2;
Jy = (1/4)*M*R^2 + (1/12)*M*thick^2;
```

```
Jz = Jy;
%Jxyz = [Jx; Jy; Jz];

J = diag([Jx Jy Jz]);
Jinv = inv(J);                          %Inverse of angular momentum modulis

I = diag([1 1 1]);

Pjx0 = Jx*Wx0;
Pjy0 = Jy*Wy0;
Pjz0 = Jz*Wz0;
Pj0 = [Pjx0; Pjy0; Pjz0];               %initial angular momentum

xo=0; yo=0; zo= R*cos(theta0);
xco=[xo; yo; zo];                       %location of center of mass

tol = 1e-10;                            %Set computational tolerances
OPTIONS = ODESET('AbsTol', tol);

%Call ode45 to compute results & load results into arrays
clear t Pjx Pjy Pjz psi theta phi xc yc zc
[t,x] = ode45('EulorDiskDot',[0 20], [Pj0' alpha0' xco'], OPTIONS);

'Begin array loading'
Pjx = x(:,1);
Pjy = x(:,2);
Pjz = x(:,3);
psi = x(:,4);
theta = x(:,5);
phi = x(:,6);

xc=x(:,7);
yc=x(:,8);
zc=x(:,9);

%Generate plots
close all
```

```
figure(1); subplot(3,3,1); plot(t,[psi]); grid;
xlabel('Time (s)');ylabel('Psi [rad]    (roll)');

figure(1); subplot(3,3,2); plot(t,[theta]); grid;
xlabel('Time (s)');ylabel('theta [rad]    (tilt)');

figure(1); subplot(3,3,3); plot(t,[phi]); grid;
xlabel('Time (s)');ylabel('phi [rad]    (pitch)');

figure(1); subplot(3,3,4); plot(t,[Pjx]); grid;
xlabel('Time (s)');ylabel('Pjx [kg.m2/s]    (roll) ');

figure(1); subplot(3,3,5); plot(t,[Pjy]); grid;
xlabel('Time (s)');ylabel('Pjy [kg.m2/s]    (tilt) ');

figure(1); subplot(3,3,6); plot(t,[Pjz]); grid;
xlabel('Time (s)');ylabel('Pjz [kg.m2/s]    (pitch) ');

figure(1); subplot(3,3,7); plot([xc],[yc]); grid;
xlabel( 'Xc [m]');ylabel('Yc [m]');

figure(1); subplot(3,3,8); plot(t,[zc]); grid;
xlabel('Time (s)');ylabel('Zc [m]');
```

```
% EulorDiskDot.m
% For Eulor disk simulation

% by: Zhaohong Wu
% 09/09/2006

function rate = EulorDiskDot(t,x)

%%% Allow sharing of system parameters
global Jinv Fxyz FXYZ tao g R thick rho_s M I

Pjx = x(1);
Pjy = x(2);
Pjz = x(3);

psi = x(4);
theta = x(5);
phi = x(6);
xc=x(7); yc=x(8); zc=x(9);

Pj = [Pjx; Pjy; Pjz];
alpha = [psi; theta; phi];

% psi matrix
psiMat = [  1              0              0
            0              cos(psi)       -sin(psi)
            0              sin(psi)       cos(psi)];

%theta matrix
thetaMat = [  cos(theta)     0          sin(theta)
              0              1          0
              -sin(theta)    0          cos(theta)];

%phi matrix
phiMat = [  cos(phi)         -sin(phi)        0
            sin(phi)         cos(phi)         0
            0                0                1];
```

```
% Matrix for transmition between Wxyz and Vxyz (body fixed)
T = [    0                  R*cos(psi)          -R*sin(psi)
        -R*cos(psi)         0                    0
         R*sin(psi)         0                    0 ];

% Matrix for trans between alphadot and Wxyz
alphaMat = [    1       sin(psi)*sin(theta)/cos(theta)     cos(psi)*sin(theta)/cos(theta)
                0       cos(psi)                           -sin(psi)
                0       sin(psi)/cos(theta)                cos(psi)/cos(theta) ]    ;

%%% Intermediate calculations

Wxyz = Jinv*Pj;
alphadot = alphaMat*Wxyz;
psidot = alphadot(1);

Fxyz = psiMat'*thetaMat'*phiMat'*FXYZ;

H = [ 0          Pjz         -Pjy
      -Pjz       0            Pjx
       Pjy      -Pjx          0     ];

Vxyz = T*Wxyz;
Pxyz = M*Vxyz;

P = [  0              Pxyz(3)        -Pxyz(2)
      -Pxyz(3)        0               Pxyz(1)
       Pxyz(2)       -Pxyz(1)        0          ];

Tdot = [    0              -R*sin(psi)      -R*cos(psi)
            R*sin(psi)      0                0
            R*cos(psi)      0                0 ];

%taox = 0;
%taoy = 0;
%taoz = -0.05;
```

```
%tao = [taox; taoy; taoz];                    %Torque applied

% Setup Ground Friction
fric1 = [ 0      0      0
          0      0      0
          0      0      0];

% fric1 = [ 0      0              0
%           0      -0.0005    0
%           0      0              -0.0005 ];

% Setup Air Dissipation
fric2 = [ 0      0              0
          0    -0.0005        0
          0      0            -0.0005 ];

% fric2 = [  0      0      0
%            0      0      0
%            0      0      0 ];

WXYZ = phiMat*thetaMat*psiMat*Wxyz;
tao1 = psiMat'*thetaMat'*phiMat'*fric1*WXYZ;    % Ground Friction

tao2 = fric2*Wxyz;      % Air Dissipation

%tao = tao1 ;
tao = tao2 + tao1;

%%% Calculate rates for each state
Pjdot = inv(I + M*T'*T*Jinv)*(-M*T'*Tdot*psidot*Wxyz - H*Wxyz + tao + T'*Fxyz
-T'*P*Wxyz);

Vc = phiMat*thetaMat*psiMat*Vxyz;
rate = [Pjdot' alphadot' Vc']';
```

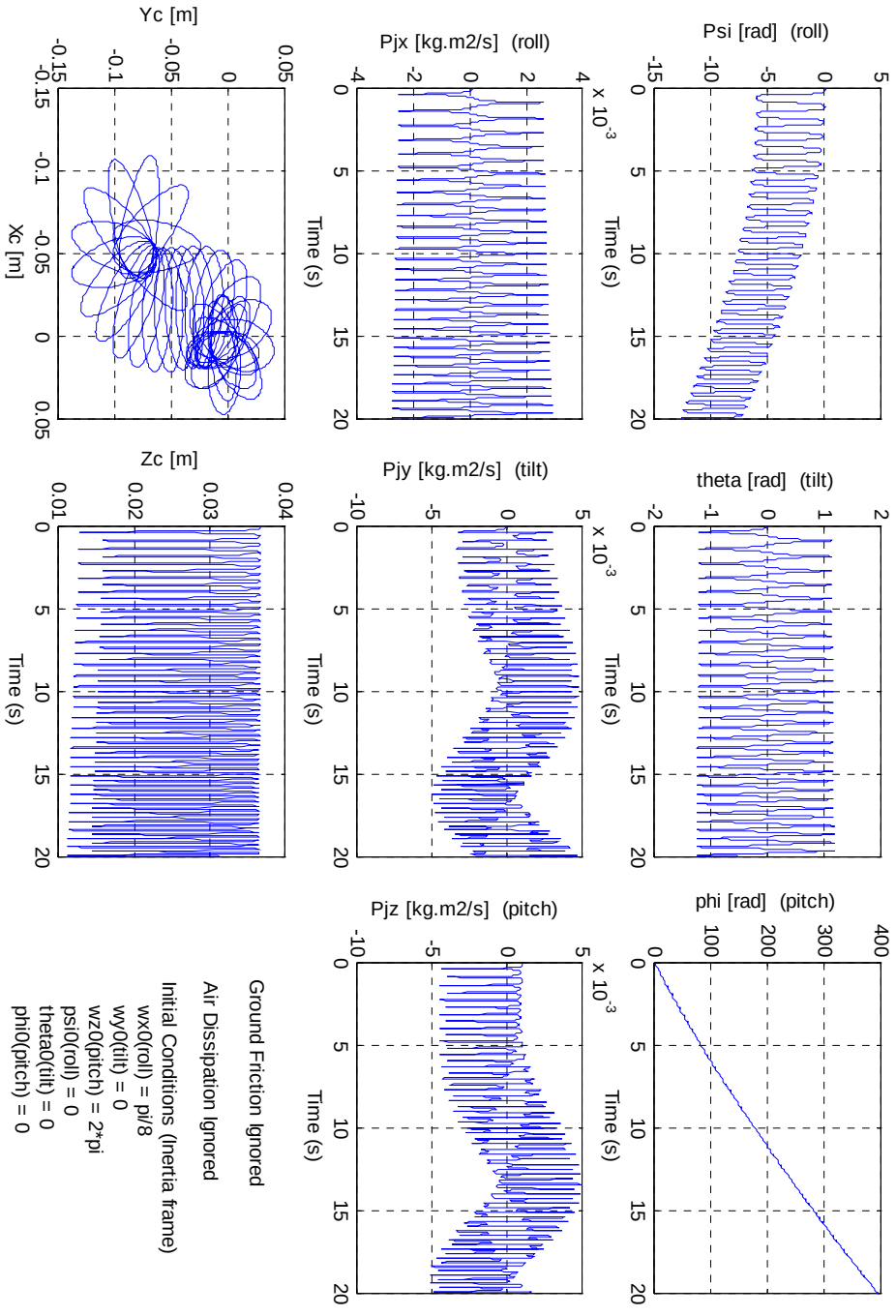


Figure C.1: Simulation of Euler Disk without air dissipation

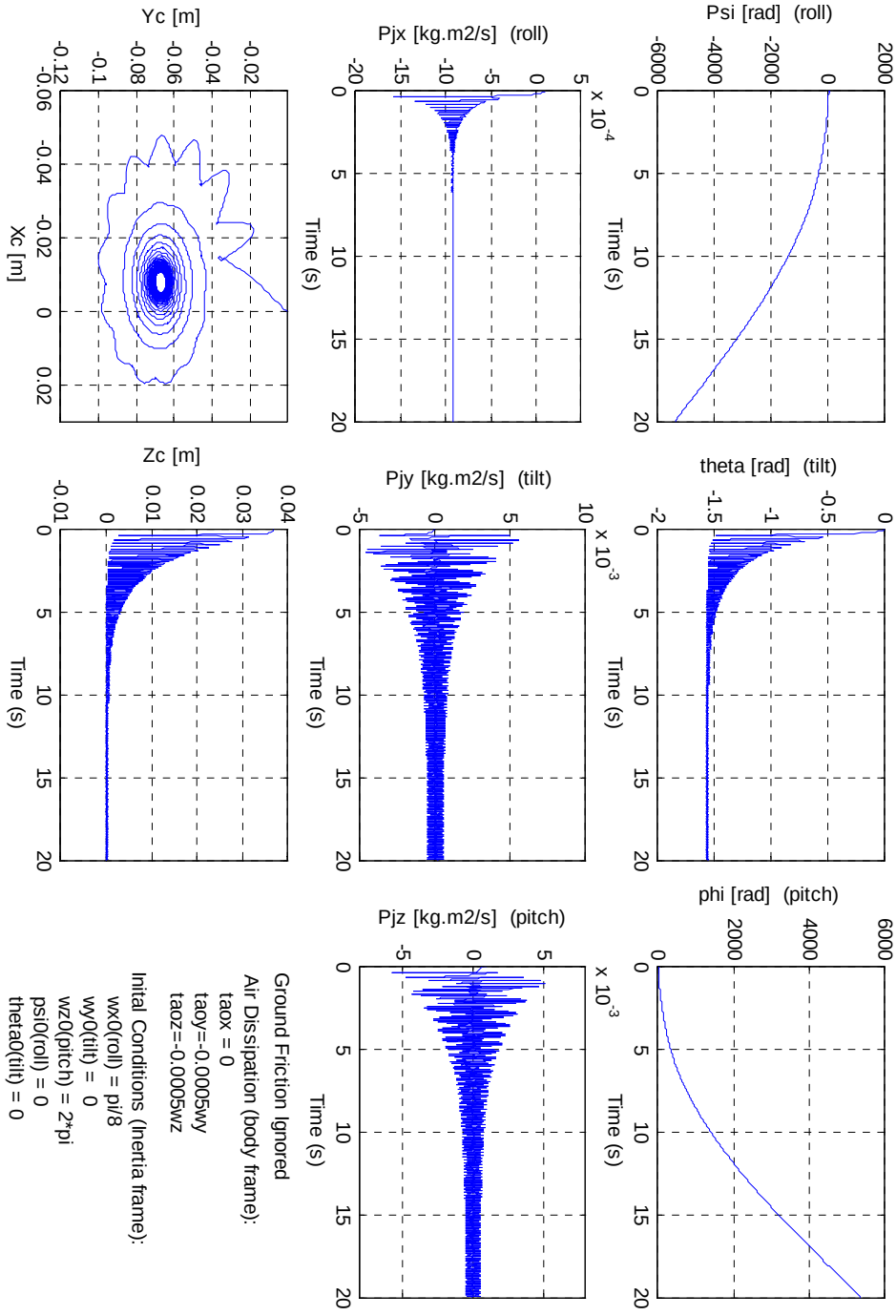


Figure C.2: Simulation of Euler Disk with air dissipation

D: OTHER PERFORMANCE METRICS FOR MODEL ORDER DERIVATION

Metric by Converging Eigenvalues

In the case of MODA, an increase in the order of a proper model generally results in some shifting of $|\lambda_i| \in (0, w_{\max})$, which suggests that a more accurate prediction can be obtained by increasing the order of the proper model. Ferris et al. [1994] addressed the issue of eigenvalue migration by creating a new model synthesis algorithm that monitors the migration of the $|\lambda_i| \in (0, w_{\max})$, which they refer to as the critical system eigenvalues. The new algorithm EXTENDED-MODA synthesizes a proper model in the same manner as MODA; it then continues increasing model order until the $|\lambda_i| \in (0, w_{\max})$ remain approximately the same as the model order is increased. The degree of approximation can be controlled by a user-specified tolerance defining the acceptable percentage change. The claim by Ferris is that EXTENDED-MODA will synthesize a model of appropriate complexity that provides estimates of $|\lambda_i| \in (0, w_{\max})$ that have converged to some user-specified percentage.

Metric by Frequency-Response

Wilson et al. [1995] developed frequency-domain model order deduction algorithm (FD-MODA) as an improvement over the selection of proper models of linear systems. FD-MODA uses the convergence of the frequency response of a system as a means of identifying its proper model. The frequency response of a system, denoted as $G(jw)$, is a useful performance metric and design aid. When plotted on the complex

plane the frequency response is the basis for Nyquist stability analysis, and is used to compute gain and phase margins. When the magnitude and phase are plotted separately, as in Bode plots, frequency response is used for frequency-domain based compensator design. We believe that considering a model's frequency response over a given frequency range provides a broader measure of its performance than monitoring eigenvalues within this same range. Frequency response is obtained by evaluating a transfer function, a ratio of polynomials, over a range of frequencies, i.e.

$$G(jw) = \frac{N(jw)}{D(jw)} \qquad \text{for} \quad w \in [w_{\min}, w_{\max}] \tag{D-1}$$

The frequency response function (D.1) is obtained from the state matrices using the well-known relation

$$G(s) = C(sI - A)^{-1}B + D \tag{D-2}$$

and substituting s = jw, where w is the input frequency in rad/sec. If we focus (for the present discussion) on SISO systems, we can rewrite (D.1) as :

$$G(s) = K\frac{\prod_{i=1}^{m}(s + z_i)}{\prod_{i=1}^{n}(s + p_i)} \tag{D-3}$$

where $m \le n$ and $m < n$ if D = 0. The poles p_i of (D.3) are the eigenvalues of the system. We now consider (D.3) in the context of the model order deduction algorithms described in the previous and next sections. Comparing the frequency response of (D.3) over a FROI $[w_{\min}, w_{\max}]$ provides a more meaningful basis for deciding submodel rank and evaluating model performance than just focusing on the poles.

There are several reasons for this assertion:

1) Using eigenvalue convergence as a criterion for setting model order may have little bearing on the convergence of $G(jw)$. For example, the variation of $G(jw)$ for a 10% change in a real eigenvalue may be quite small, while a 10% change in a pair of very lightly damped eigenvalues may have a major impact.
2) In the context of control system analysis and design, $G(jw)$ tells the "whole story" regarding gain and phase margin, loop-shaping requirements (for compensator design), and the like; eigenvalues convey only part of this information.

In the final analysis, we want the system model to provide a reliable prediction of model performance for $w \in [w_{\min}, w_{\max}]$. For this to occur, both the zeros and the poles of (D.3) must have converged sufficiently such that increases in model order do not cause appreciable changes in the frequency response. This simply cannot be done by considering only the eigenvalues.

REFERENCE

Balance, D. J., Bevan, G. P., Gawthrop, P. J. and Diston D. J., 2005, "Model Transformation Tools (MTT): The Open Source Bond Graph Project," International Bond Graph Modeling Conference, New Orleans, LA.

Baluja, S., 1994, "Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning", Technical Report CMU-CS-94-163, Carnegie Mellon University.

Baluja, S., and Caruana, R., 1995, "Removing the Genetics from the Standard Genetic Algorithm," in Proceedings of the International Conference on Machine Learning.

Baluja, S., 1996, "Genetic Algorithms and Explicit Search Statistics," Advances in Neural ndformation Processing Systems (NIPS '96).

Baluja, S., 1997, "Genetic algorithms and explicit search statistics", Advances in Neural Information Processing Systems 9, Proceedings of the 1996 Conference, pp.319.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. 1998. Genetic Programming – An Introduction. San Francisco, CA: Morgan Kaufmann and Heidelberg.

Beaman, J.J., and Paynter, H.M., 1995, "Modeling of Physical Systems," (unpublished manuscript), University of Texas at Austin, TX.

Bohm, M., and Stone, R., 2004, "Representing Functionality to Support Reuse: Conceptual and Supporting Functions," Proceedings of DETC'04, DETC2004-57693, Salt Lake City, UT.

Bohm, M. and Stone, R., 2003, "Refining Design Repositories: Creating a Usable Framework with XML Data Representation," Proceedings of the 2003 NSF Grantees Conference, Birmingham, AL.

Borutzky, W., 1999, Bond graph modeling from an object oriented modeling point of view, Simulation Practice Theory, 7(5-6), pp 439-461.

Breedveld, P.C., 1988, "Bond graph-based Model Generation," Automated Modeling for Design, ASME Winter Meeting, DSC Vol. 8, pp. 41-417.

Campbell, M.I., 2000, “The A-Design Invention Machine,” PhD Dissertation, Carnegie Mellon University.

Campbell, M.I., J. Cagan and K. Kotovsky, 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Research in Engineering Design, Vol. 11, No. 3.

Castillo, O. and P. Melin, 1999, “Automated mathematical modeling, simulation and behavior identification of robotic dynamic systems using a new fuzzy-fractal-genetic approach,” Robotics and Autonomous Systems, Volume 28, Issue 1, 31, pp. 19-30.

Cohoon, J., Hedge, S., Martin, W., Richards, D., 1988, “Distributed Genetic Algorithms for the Floor Plan Design Problem,” School of Engineering and Applied Science, Computer Science Dept., University of Virginia, TR-88-12.

De Jong, K., 1975, An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Dissertation.

Easwar, K., Rouyer, F., and Menon N., 2002, "Speeding to a stop: The finite-time of a spinning disk," Physical Review, E 66, 045102(R), The American Physical Society.

Elmqvist, H., et al. ModelicaTM – a unified object-oriented language for physical systems modeling. HTTP://WWW.MODELICA.ORG.

Eryilmaz, B., 1996, Modeling, analysis, and control of single-input single-output systems with uncertain physical parameters, Master's thesis, Northeastern University.

Eshelman, L.J., 1991, “The CHC Adaptive Search Algorithm,” in Rawlings (ed.) Foundations of GAs-1. pp. 265-283.

Ferris, J.B., Stein, J.L. and M.M. Bernitsas. 1994. "Development of Proper Models of Hybrid Systems". 1994 ASME Winter Annual Meeting, Proceedings of the Symposium Automated Modeling for Design. (Chicago, II, Nov.7-11). ASME, New York, NY.

Finger, S., and Rinderle, J.R., 1989, "A Transformational Approach to Mechanical Design Using A Bond graph Grammar," Design Theory and Methodology - DTM '89, DE Vol. 17, pp. 107-116.

Fontana W. and Buss, L. W., 1994, " `The Arrival of the Fitest': Towards a Theory of Biological Organization," Bulletin of Mathematical Biology 56: 1—64.

Franklin, G.F. and Powell, J.D., 1991, Feedback Control of Dynamic Systems, Addison-Wesley, Reading, MA.

Free Software Foundation (FSF), 1991, "GNU general public license," [Online]. Available: http://www.gnu.org/copyleft/gpl.html

Gawthrop, P. J., 1995, "MTT: Model transformation tools," in Proceedings of the international conference on Bond Graph Modeling and Simulation (ICBGM' 95), Las Vegas, NV.

Gawthrop, P. J. and Bevan G. P., 2003, "MTT: Model transformation tools – manual for version 5.0," Technical Report, 1003. [Online].Available: HTTP://PRDOWNLOADS.SOURCEFORGE.NET/MTT/MTT.PDF?DOWNLOAD

Gawthrop, P. J. and Smith L. P. S., Metamodelling: Bond Graphs and Dynamic Systems. Hemel Hempstead, Herts, England.: Prentice Hall, 1996.

Goldberg, D.E., 1989, Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Goldberg, D.E., Richardson, J, 1987, "Genetic Algorithms with Sharing for Multimodal Function Optimization," Proceedings of the Second International Conference on Genetic Algorithms.

Goldberg, D.E., Segrest, P., (1987) Finite Markov chain analysis of genetic algorithms. Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, 1-8.

Harik, G.R., Lobo, F.G., Goldberg, D.E., 1999, "The Compact Genetic Algorithm." IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4 (1999) 287-297.

Hornby, G.S., and Lipson, H., and Pollack, J.B., 2003, "Generative Representations for the Automated Design of Modular Physical Robots." IEEE transactions on Robotics and Automation, 19(4):709-713.

Houck, C., Joines, J., and Kay, M., 1995, "A Genetic Algorithm for Function Optimization: A Matlab Implementation," NCSU-IE TR 95-09.

Holland, J. 1992, "Genetic Algorithms", Scientific American, Easy introduction by the father of the field.

Kaelbling, L.P., Littman, M.L., and Moore, A.W., 1996, "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research, Vol. 4, pp. 237-285.

Karnopp, D.C., Rosenberg, R.C., 1983, Introduction to Physical System Dynamics, USA: McGraw-Hill.

Karnopp, D.C., Margolis, D.L., and Rosenberg, R.C., 1990, System Dynamics: A Unified Approach, John Wiley and Sons, New York.

Karnopp, D.C., Margolis, D.L., and Rosenberg, R.C., 2000, System Dynamics: Modeling and Simulation of Mechatronic Systems, John Wiley and Sons, New York. 3rd edition.

Kohonen, T., 1989, "Self-organization and Associative Memory (3rd Ed.)", Berlin, Springer-Verlag.

Koza, J. R. 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press.

Koza, J. R.: 1994a, Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, Cambridge, MA, USA

Koza, J.R., 1994b, Genetic Programming II Videotape: The Next Generation. MIT Press, Cambridge, MA, USA

Koza, J.R., Bennett F.H., Forrest, H., Andre, D., and Keane, M.A., 1999, Genetic Programming III: Darwinian Invention and Problem Solving. San Francisco, CA: Morgan Kaufmann.

Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A., Dunlap, F., 1997, "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," IEEE Transaction on Evolutionary Computation, vol. 1, no. 2, pp. 109~128.

Koza, J.R., and Rice, J.P., 1992, Genetic Programming: The Movie. Cambridge, MA: MIT Press.

Kurtoglu, T., Campbell, M.I., Gonzalez, J., Bryant, C. R., Stone, R. B., McAdams, D. A., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," ASME 2005 International Design Engineering and Technical Conference and Computers and Information in Engineering Conferences, Long Beach, CA

Laird, J.E., Newell, A., and Rosenbloom, P.S., 1986, "Soar: An Architecture for General Intelligence," Technical Report CMU-CS-86-171, Carnegie Mellon Univ., Pgh., PA.

Lenat, D.B., 1983, "EURISKO: A Program That Learns New Heuristics and Domain Concepts", Artificial Intelligence, Vol. 21, pp. 61-98.

Louca, L.S. and J.L. Stein, 1999. "Energy-Based Model Reduction of Linear Systems". 1999 International Conference on Bond Graph Modeling and Simulation, San Francisco CA. Not included in conference proceedings.

Michalewicz, Z., 1994, "Genetic Algorithms + Data Structure = Evolution Programs," AI Series. Springer-Verlag, New York.

Mühlenbein, H., 1998, “The Equation for Response to Selection and Its Use for Prediction. Evolutionary Computation,” Vol. 5, 303-346.

NIST Workshop on Product Representation for Next-Generation Distributed Product Development, National Institute of Standards and Technology, Gaithersburg, MD, Nov. 30 - Dec. 1, 2000.

Papalambros, P.Y., and Wilde, D.J., 2000, “Principles of Optimal Design,” Cambridge University Press, New York.

Patel, J., Campbell, M.I., 2005, “Automated Synthesis of Sheet Metal Parts by Optimizing a Fabrication Based Graph Topology,” 1st AIAA Multidisciplinary Design Optimization Specialist Conference. Paper No. 2207, April 18-21, Austin, TX.

Paynter, H.M., 1961, “Analysis and Design of Engineering Systems, MIT Press, Cambridge, MA.

Pelikan, M., Goldberg, D.E., Cantú-Paz, E., 1998, “Linkage problem, distribution estimation, and Bayesian networks.” IlliGAL Report No. 98013: Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.

Rai, R. and Campbell, M.I., 2006,"Qualitative and quantitative sequential sampling," Proceedings of Design Engineering Technical Conference, Pennsylvania, USA

Rao, S. S., 1990, Mechanical Vibrations, Addison-Wesley Publishing Company, second edition.

Rosenberg, R.C., 1975, "The Bond graph as a Unified Database for Engineering System Design," Journal of Engineering and Industry, Transactions of ASME, Vol. 97, No. 4, pp. 1333-1337.

Rosenberg, R.C., Goodman, E.D., and Seo, K., 2001, “Some Key Issues in Using Bond Graphs and Genetic Programming for Mechatronic System Design” ASME IMECE conference, New York, NY.

Ruiz, J.O., Raju, S., and Fernández, B., 1995, "Neural Network Synthesis using Genetic Algorithms," 1995 Artificial Neural Networks in Engineering, ANNIE '95, St. Louis, MO, November 12–15.

Sacks, E. and L. Joskowicz, 1993, “Automated modeling and kinematics simulation of mechanisms,” Computer-Aided Design, Volume 25, Issue 2, pp 106-118, February 1993.

Sandholm, T.W., and Crites, R.H., 1995, “On Multiagent Q-learning in a Semi-Competitive Domain”, Adaptation and Learning in Multi-Agent Systems. IJCAI 95 Workshop. Proceedings, eds. G. Weiss, S. Sen., Vol. 6, 191-205

Sendur, P., Stein, J.L., Louca, L.S., and Peng H., 2003, “An Algorithm for the assessment of reduced dynamic system models for design”. Proceedings of the International Conference on Simulation and Multimedia in Engineering Education, pp. 92-101, Orlando, FL. Published by the Society for Modeling and Simulation International Simulation, ISBN 1-56555-261-1, San Diego, CA.

Seo, K., Hu, J., Fan, Z., Goodman, E. D., and Rosenberg, R. C., 2002, "Automated Design Approaches for Multi-Domain Dynamic Systems Using Bond Graphs and Genetic Programming,," The International Journal of Computers, Systems and Signals, vol.3, no.1, pp.55-70.

Sharpe, J.E. and Bracewell, R.H. 1995. “The use of bond graph reasoning for the design of interdisciplinary schemes.” Proceedings of the 1995 international Conference on Bond Graph Modeling. Pg. 116-121, January, Las Vegas, NV. SCS, San Diego, CA.

Stein, J.L., and L.S. Louca, 1995, "A Component-Based Modeling Approach for System Design: Theory and Implementation," Proceedings of the 1995 International Conference on Bond Graph Modeling and Simulation, Vol. 27, No, 1, pp. 109-115, Las Vegas, NV. Published by SCS, ISBN 1-56555-037-4, San Diego, CA.

Stein, J.L, Louca, L.S., 1996, “A template-based modeling approach for system design: Theory and implementation, Transactions of the society for computer simulation international. Published by SCS ISSN 0740/679796, San Diego, CA.

Szykman, S., Racz, J., and Sriram, R., 1999, “The Representation of Function in Computer-Based Design,” Proceedings of DETC99, DETC99/DTM-8742, Las Vegas, NV.

Taylor, J.H. and Wilson, B.H., 1995, “A frequency domain model-order deduction algorithm for nonlinear systems,” IEEE Conference on Control Applications, Albany, NY.

Tan, M., 1993, “Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents”, Tenth International Conference on Machine Learning, Amherst, MA, pp. 330-337.

Tay, E., Flowers, W., and Barrus, J., 1998, “Automated Generation and Analysis of Dynamic System Design,” Research in Engineering Design, Vol. 10, pp. 15-29.

Ulrich, K., and Seering, W., 1989, “Synthesis of Schematic Descriptions in Mechanical Design,” Research in Engineering Design, Vol. 1, pp. 3-18.

Welch, R. V., and Dixon, J., 1994, “Guiding Conceptual Design Through Behavioral Reasoning,” Research in Engineering Design, Vol. 6 pp. 169-188.

Whitley, D., & Starkweather, T. “Genitor II: A Distributed Genetic Algorithm”. JETAI 2: 189-214.

Wilson, B.H. and Stein J.L., 1992, “An Algorithm for Obtaining Minimum Order Models of Distributed and Discrete An Approach for System Development Using Evolutionary Probabilistic Strategy and Grammar Rules ASME Winter Annual Meeting. Symposium on Automated Modeling. (Anaheim, CA, Nov. 8-13). ASME Book No. G00747, ASME, New York, NY.

Wilson, B.H., Taylor, J.H., and Eryilmaz, B., 1995, “A frequency domain model order deduction algorithm for linear systems,” In T. E. Alberts, editor, Proceedings of the ASME Dynamic Systems and Control Division, volume DSC-Vol. 57-1, pages 401-410, San Francisco, CA. ASME International Mechanical Engineering Congress and Exposition.

Wilson, B.H., Stein, J., 1995, “An algorithm for obtaining minimum order models of distributed and discrete systems,” Journal of Dynamic Systems, Measurement, and Control,117(4).

Winston, P. H., 1982, “Learning New Principles from Precedents and Exercises,” Artificial Intelligence, vol.19, no.3, p. 321-350.

Winston, P. H., 1992, Artificial Intelligence, Addison-Wesley, 3rd edition, Reading, MA.

Wu, Z., Fernandez, B., Campbell, M., 2006, "An Approach for System Development Using Evolutionary Probabilistic Strategy and Grammar Rules", ASME Artificial Neural Networks in Engineering conference, St. Louis, MS.

Wu, Z., Campbell, M., Fernandez, B., 2007, "Bond Graph Based Automated Modeling for Computer Aided Design of Dynamics Systems”, Accepted to Journal of Mechanical Design.

Vita

Zhaohong Wu was born in Lianyuan, Hunan Province in the middle south of China on July 6th, 1975 to Zhiping Wu and Peixuan Liu. Following parents, he grew up in the very northeast of China before He went to Beijing for college. He received his B.E. and M.E. degrees in Manufacturing Engineering from Beijing University of Aeronautics and Astronautics in September 1997 and March 2000 respectively. He started his PhD in Mechanical Engineering department at UT, Austin in June 2001. He took half a year leave during his study due to visa difficulty and was readmitted to UT in June 2003. He met his wife Yanjing Zhou in Austin in 2004 when she came for her MBA study. They got married in 2005 and had their lovely daughter Kristin Zhou Wu in 2006. During his study at UT, he published over 10 papers in conference proceedings and academic journals.

Permanent address: Building 3, #412, Hao Zhuang Jia Yuan,
Changping District, Beijing, China, 102200

This dissertation was typed by the author.