DESIGN CONSIDERATIONS FOR A RECONFIGURABLE DELTA ROBOT

BY

JEFFREY D. ARENA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Systems and Entrepreneurial Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Assistant Professor James T. Allison

ABSTRACT

One project, the design of a reconfigurable Delta robot, motivated three different works which comprise this thesis. In Chapters 1-2, I discuss the application of multiobjective design principles to decide appropriate ranges of reconfigurability in a Delta robot. I show that the selection reconfigurable ranges in a system designed for multiability through continuous reconfiguration can be accomplished by generating points along an "expanded Pareto set"– each of these points representing a fully reconfigurable design. In Chapters 3-4 I discuss dynamically modeling compliance in a structure, and use the derivation to formulate the co-design problem for a simple pick-and-place manipulator made up of one compliant link. After displaying existing results, I discuss the current state of this research and potential next steps. Finally, in Chapter 5, I outline the practical considerations that comprised my construction of a reconfigurable Delta robot.

To Emily,
for her love and support.

And to Christian Burns Barden,
the newest addition to my family.

Professor James T. Allison was the deciding factor in my decision to attend the University of Illinois. Two years later, I know I made the right choice. As an adviser, professor Allison always took the time to understand my interests, and focus my enthusiasm into useful research contributions. Without his support and insightful advice, I would not be the graduate researcher that I am today.

While working under professor Allison in the Engineering System Design Lab, I have been able to work with a great group of graduate students. Allen Kaitharath, Anand Deshmukh, Ashish Khetan, Dan Herber, Jason McDonald, Lakshmi Rao, and Tinghao Guo—thank you laughing at my jokes (usually) and for making even the most stressful times in the lab enjoyable.

Outside of the ESDL, I was incredibly lucky to have met and worked with Dan Block, an academic professional who gave me hands-on experience to supplement research. Originally through Mechatronics class and later working for Dan as a lab developer for the College of Engineering Control Systems Lab (COECSL), he has taught me so much as an engineer. Through countless projects, Dan has always projected a positive attitude while relentlessly pushing deeper understanding of applied control design. The skills I have developed under his guidance will continue to be invaluable in my future engineering career.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$\bar{\mathbf{x}}$	(Design for Reconfigurability) Cost function, measures cost of reconfigurability based on reconfigurability design vector $\bar{\mathbf{x}}$
$\boldsymbol{\Xi}$	(Direct Transcription) Discretized version of state trajectory. The ith row of $\boldsymbol{\Xi}$ is the state vector at the ith timestep.
$\boldsymbol{\xi}(t)$	(Direct Transcription) The vector of system states as a function of time
$\boldsymbol{\zeta}(\cdot)$	(Direct Transcription) Defect constraints. If $\boldsymbol{\zeta}(\mathbf{U}, \boldsymbol{\Xi}) = \mathbf{0}$ then the discretized version of the system dynamic equations is satisfied.
η_{ij}^{pq}	(Product Family Design) binary variable (i.e. $\eta_{ij}^{pq} \in \{0, 1\}$) indicating if components i and j are shared between products p and q
$\mathbb{X}$	the set of all feasible design vectors.
$\boldsymbol{\Gamma}$	(Reconfig. Delta Case Study) Upper masking matrix in Lipkin and Patterson's alternative eigenproblem $\left(\boldsymbol{\Gamma} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}\right)$
$\boldsymbol{\Omega}$	(Reconfig. Delta Case Study) Lower masking matrix in Lipkin and Patterson's alternative eigenproblem $\left(\boldsymbol{\Omega} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}\right)$
$\overline{\mathbf{x}}$	(Design for Reconfigurability) For m objectives, and n design variables, it is made up of mn elements, the first n are the design variables for the first objective function f^1, the next n are the design variables for the second objective function f^2, etc.
$\mathbf{f_d}(\cdot)$	(Direct Transcription) The state space time derivative function. The derivative of the system states is given as $\dot{\boldsymbol{\xi}} = \mathbf{f_d}(\boldsymbol{\xi}(t), \mathbf{u}(t), \mathbf{x}_p, t)$

$\mathbf{F}$	Vector-valued function with ith element f^i
$\mathbf{g}^p$	(Product Family Design) inequality constraints in product p
$\mathbf{h}^p$	(Product Family Design) equality constraints in product p
$\mathbf{p}$	(Goal Programming) Vector of positive deviations from aspirations— element i referred to as p_i
$\mathbf{r}$	(Goal Programming) Vector of negative deviations from aspirations— element i referred to as r_i
$\mathbf{U}$	(Direct Transcription) Discretized version of control trajectory. The ith row of $\mathbf{U}$ is the control applied at the ith timestep.
$\mathbf{u}(t)$	(Direct Transcription) The vector of controls applied to a system as a function of time.
$\mathbf{x}$	Vector of design variables
$\mathbf{x}^*$	An optimal (usually minimizing) design vector
$\mathbf{x}^i$	(Design for Reconfigurability) Design vector for task i, containing its n design variables
$\mathbf{x}^p$	(Product Family Design) Design vector for product p
$\mathbf{x}_p$	(Direct Transcription) The vector of plant design variables
$\mathcal{P}$	(Product Family Design) Set of indices, each referring to a product p
ν	(Control of Compliant Link) Rayleigh dissipation coefficient (N·s/m)
$A(\mathbf{p}, \mathbf{r})$	(Goal Programming) Achievement function— assigns a scalar value to a set of positive and negative deviations from aspirations
a_i	(Goal Programming) Aspiration for objective i
b	(Control of Compliant Link) Link out-of-plane thickness (m)
c_i	Scalar upper bound for objective function i (f^i) in ϵ-Constraint multiobjective optimization approach
D	(Reconfig. Delta Case Study) Vertical bar separation distance (m)

E	(Reconfig. Delta Case Study) Material elastic modulus (m)
f^i	ith objective function, $f^i : \mathbb{R}^n \mapsto \mathbb{R}$
f^n_{target}	Target value, or "ideal value" of the nth objective function used to calculate a distance in the objective space in compromise programming.
f^p	(Product Family Design) Performance function for product p
h	(Control of Compliant Link) Link in-plane thickness (m)
h_i	(Direct Transcription) The time duration of the ith timestep
L	(Reconfig. Delta Case Study) Strut length (m)
$L(\cdot)$	(Direct Transcription) The running cost of a particular control and state trajectory. The total cost of the trajectory is calculated by integrating this quantity over time as in $\int_0^{t_F} L(\boldsymbol{\xi}(t), \mathbf{u}(t), \mathbf{x}_p)dt$
M_{end}	(Control of Compliant Link) Mass of payload (Kg)
R_{max}	(Reconfig. Delta Case Study) Maximum build area radius used for design metric calculation (m)
S	(Reconfig. Delta Case Study) Base triangle side length (m)
S^{pq}	(Product Family Design) Index pairs of elements that are candidates for being shared between two products p and q
T	(Reconfig. Delta Case Study) Strut diameter (m)
u_{max}	(Control of Compliant Link) Control saturation value (N·m)

FOREWORD

The goal of my Master's degree was to write two papers to reputable conferences, and learn as much as possible in two years. My interests have not changed since entering graduate school. They are design optimization, control, and robotics. It is in these areas that I sought to make contributions to existing literature. I have tried to develop design methods that are both practical and mathematically rigorous.

This work is composed of three main parts which are small subsets of the three subjects I sought to learn during my Master's degree. These are: multiobjective optimization (Chapters 1 and 2), dynamic system design (Chapters 3 and 4), and robotic implementation and fabrication (Chapter 5). In Chapter 1, I give a concise background for multiobjective optimization. In Chapter 2, I dive deep into an application of multiobjective optimization: reconfigurable system design. In Chapter 3, I give an introduction to some theory behind designing dynamic systems, introducing methods that I used for the research presented in Chapter 4. In Chapter 5, I describe the design and fabrication of a working and reconfigurable delta robot. The material in Chapter 5 is on a practical rather than theoretical level.

The material for the first two major parts of this work is drawn heavily from my publications. The thesis format gives me much more freedom to introduce the rich background which each research work is based on. Additional content included in this document includes derivations and general background information about the subject area.

All research and fabrication essentially took place in parallel. I would love to say that I fully incorporated the results of all my papers into the design of my delta robot, but unfortunately this was not possible. In the future, I hope to take the lessons learned from each of these efforts to build a truly optimal reconfigurable, dynamic system. For now, I detail the steps required to arrive at the current phase of research and fabrication.

CHAPTER 1

MULTIOBJECTIVE OPTIMIZATION

1.1 Introduction

Almost always, real decisions require the consideration of more than one objective. Should a city build its firestations nearer to its heavily populated residential districts or its high value business districts [3]? How much should a car company be willing to sacrifice fuel efficiency to provide more cargo room to its customers? How much should a person be willing to pay to reduce the chances of their death?

All of these decisions contain trade-offs between competing objectives. And even when these objectives can be quantified, selecting an "optimal" answer can be difficult—requiring a decision-maker to survey possible solutions and make value judgments between alternatives. In the design of engineering systems, we call the decision-maker the designer. Their decisions are product specifications, and their objectives may include cost, performance, robustness, or even aesthetics. The field of multi-objective optimization (MOO) attempts to aid the designer by breaking down complicated problems into a mathematical formulation before finding a select group of the most desirable designs for the given criteria.

The next several sections are a general description of MOO from the designer's perspective. First, I outline the generic problem structure in MOO along with some typical terminology used. Next, I detail some common techniques used in practice to find acceptable solutions to these problems. After this introduction, an example of multiobjective objective optimization is presented involving the design of a reconfigurable delta robot.

General Multiobjective Optimization Formulation

The general form of a multiobjective design problem may be given as:

$$\begin{aligned} \min_{\mathbf{x}} \ & \{f^1(\mathbf{x}), ..., f^m(\mathbf{x})\} \\ \text{such that } \ & \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n \end{aligned} \tag{1.1}$$

In formulation 1.1, $\mathbf{x}$ is the vector of design variables (also called the design vector), $f^i : \mathbb{R}^n \mapsto \mathbb{R}$ is the ith of (m total) objective functions, and $\mathbb{X}$ is the set of all feasible design vectors.

Consider the vector-valued function $\mathbf{F}$ which is defined as follows:

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} f^1(\mathbf{x}) \\ \vdots \\ f^m(\mathbf{x}) \end{bmatrix} \tag{1.2}$$

Thus, the function $\mathbf{F}$ maps points in $\mathbb{X}$ into $\mathbb{R}^m$, where the ith coordinate of $\mathbf{F}(\mathbf{x})$ is given by $f^i(\mathbf{x})$.

If we let the design vector $\mathbf{x}$ vary over all $\mathbb{X}$, $\mathbf{F}(\mathbf{x})$ will vary over all attainable points in $\mathbb{R}^m$. Unsurprisingly, this set of attainable points in $\mathbb{R}^m$ is called the "attainable set." In this general formulation, we refer to the space $\mathbb{R}^n$ as the design space (which contains the feasible set), and R^m as the objective space (which contains the attainable set.)

1.1.1 The Pareto Set

Put simply, the Pareto is the set of all the "best" points in the objective space.[1] In the context of engineering design, a point in the objective space is considered one of the "best" points if there does not exist another point with a better (usually defined as lower) score in one objective, and at least equal scores in all others. The logic here is clear; if a point A in the objective space has a lower score in one objective than another point B, and equal (or lower)

[1]The name "Pareto set" originates from Vilfredo Pareto (1848-1923), an Italian engineer-turned-economist who defined a concept of optimality in the context of resource distribution in a society. Given a set of individuals in a society, and some level of resources to distribute among them, a given distribution of resources was considered "Pareto optimal" or "Pareto efficient" if no other distribution existed that improved the situation of one individual without worsening the situation for any others.

scores than B in all other objectives, then point A is objectively better than point B. Researchers say that the point A *dominates* point B. The set of all non-dominated points in the objective space is the Pareto set.

1.1.2 Example Multiobjective Optimization

The following example multiobjective problem demonstrates all of these ideas. Consider the following unconstrained minimization problem.

$$\begin{aligned} \min_{\mathbf{x}=[x_1,x_2]^T} \quad & \{f^1(\mathbf{x}), f^2(\mathbf{x})\} \\ \text{where:} \quad & f^1(\mathbf{x}) = \left(\frac{x_1 - 1}{1^2}\right)^2 + (x_2)^2 \\ & f^2(\mathbf{x}) = (x_1)^2 + \left(\frac{x_2 + 0.5}{2^2}\right)^2 \end{aligned} \tag{1.3}$$

In this case, the design space, $\mathbb{X}$, is $\mathbb{R}^2$, and the objective space is $\mathbb{R}^2$. In this example we can easily graph points along the problem's Pareto front. Five Pareto optimal points are shown in Fig. 1.1. Their preimages are shown in figure 1.2.

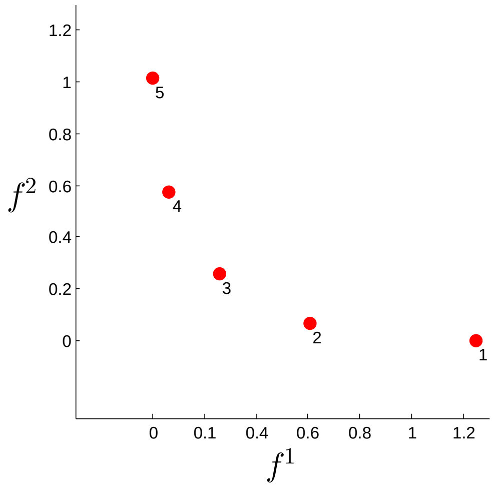


Figure 1.1: Five points along the Pareto frontier of optimization formulation 1.3

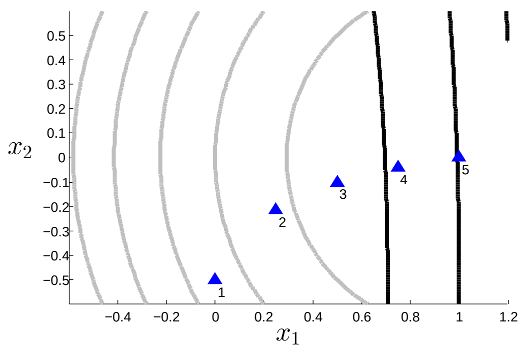


Figure 1.2: Preimage of points in figure 1.1. The solid lines (shown in two distinct shades) are level curves for the two objective functions.

1.2 Multiobjective Optimization Methods

Before outlining some of the methods that have been employed to solve MOO problems, I want to provide additional context to frame the approaches. All MOO strategies can be classified somewhere between the two following extremes: (1) generating approaches and (2) preference-based approaches [3, 4]. The goal of a generating approach is to find the Pareto frontier, and allow decision-makers to select points from this set afterward. There is no "best" way to generate an even distribution of points along the Pareto set however, and this generation can be expensive depending on the problem. So while, theoretically, generating approaches offer the designer a choice from the "best" solutions to a problem, the difficulty is in the generation of a well-distributed set of points along this set.

Preference-based methods are at the other extreme. Rather than attempting to generate a set of points along the Pareto set, preference-based methods try to capture designer preferences in a scalar function (often called a "utility function") before optimizing. The advantage of this method is that it reduces a multiobjective problem to the minimization of a single scalar objective. The difficulty, however, is in creating a utility function that truly captures the designer's preferences.

The following MOO methods can operate anywhere in between these extremes. Practical application will often call for iterative application, resulting in the generation of some optimal points along the Pareto set. However care should be taken that each subsequent iteration better captures the true preferences of the designer, so that solutions may be determined without the need for an exhaustive search.

1.2.1 Objective weighting

One of the most intuitive ways to determine an optimal solution to a multiobjective problem is using objective weighting. Each objective is assigned some positive weight which corresponds to its importance in the multiobjective problem. The generic MOO formulation given in 1.1 can be reframed as a weighted sum as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \alpha_1 f^1(\mathbf{x}) + ... + \alpha_m f^m(\mathbf{x}) \\ \text{such that} \quad & \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n \end{aligned} \tag{1.4}$$

Where α_1,..., α_m are scalar objective weights. In this way, the m different objectives in the original problem formulation are scalarized into a single value. Instinctively, it is clear that increasing the linear weight applied to a given objective, increases the objectives relative importance in the multiobjective problem.[2]

While this strategy can be attractive due to its simplicity, it has shortcomings which should not be overlooked. A designer is counting on objective weights to account for unit inconsistencies. When a designer chooses some set of weights, they imply some linear tradeoff between these objectives that might be overly simplistic. Is a home insurance buyer looking to minimize risk (f_1) and cost (f_2) always willing to get α_1 less units of risk for less than α_2 units of additional cost? If not, the method of objective weighting does not truly capture the consumer's preferences.

There is another, more theoretical shortcoming of objective weighting; it is not capable of finding points on non-convex regions of the Pareto set [5]. This deficiency is especially relevant if the objective weighting method is employed as a generating method. For example, a designer might try to discover n points along the Pareto front by solving n separate optimization problems, each with a different set of weights. But because objective weighting cannot

[2]It is fairly common practice to require the sum of the weights $\alpha_1, ..., \alpha_n$ be equal to one. This reduces the number of user-specified weights by one. Forcing the sum of weights to equal one also requires that objectives be scaled similarly.

find points on concave portions of the Pareto front, those portions of the Pareto set will remain hidden to the designer. Further, even if the Pareto set is convex, it is not clear how to choose n different sets of objective weights that will yield relatively evenly-distributed points along the Pareto frontier.

Some researchers have attempted to address these shortcomings by modifying the approach. The normal boundary intersection method [6, 7], normal constraint method [8], and adaptive-weighted sum method [9] are some examples of efforts to address these shortcomings.

1.2.2 ϵ-Constraint

The ϵ-constraint method is very simple, but is capable of finding points along non-convex regions of the Pareto frontier. Originally developed by Marglin [10], the method requires the designer to transform all objectives into constraints except for one. The levels of these constraints are fixed at some acceptable value while the remaining objective is optimized alone. We can rewrite our general formulation from 1.1 in this form easily as:

$$\min_{\mathbf{x}} \quad f^1(\mathbf{x}) \tag{1.5}$$

$$\text{such that} \quad f^i(\mathbf{x}) < c_i \quad \forall i \in \{2, 3, ..., m\} \tag{1.6}$$

$$\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n$$

Note that, without loss of generality, we have assumed f^1 is the function being minimized while all other functions f^i are held beneath some acceptable value of c_i. By varying the values of c_i and solving the resulting set of optimization problems, a designer can find points along the Pareto frontier.

1.2.3 Distance Function

The distance function method (also known as compromise programming) works by selecting a point in the objective space with minimal distance to some target point that is not attainable. It was first introduced by Zeleny in [11]. The "distance" between attainable points in the objective space and the target point is measured by using some norm. We can reformulate our general multiobjective problem in terms of a distance function with the following formulation:

$$\min_{\mathbf{x}} \quad ((f^1(\mathbf{x}) - f^1_{target})^p + ... + (f^m(\mathbf{x}) - f^m_{target})^p)^{\frac{1}{p}} \tag{1.7}$$
$$\text{such that } \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n$$

Selecting values of p from 1 to inf corresponds to a so-called efficiency-equity tradeoff. Selecting $p = 1$ reduces this formulation to minimizing the sum of the individual deviations from the target point, and implies that the sum of all objective function deviations should be minimized. This solution can be viewed as having the most overall *efficiency*. To the other extreme, letting the value of p approach infinity corresponds to solving the "minimax problem," i.e. selecting the x which minimizes the value of the maximum deviation from the target point.[3] It is easy to see why the solution to the minimax problem can be viewed as the most equitable.

1.2.4 Lexicographic

Lexicographic approaches establish a priority order between objectives, and sequentially minimize according to this order of importance. Let f^i be the ith ranked objective function in our general formulation. Solving this formulation lexicographically would mean we sequentially solve the following formulations, where we define $\mathbf{x}_i^*$ to be a solution to the ith formulation in the sequence.

[3]The infinity-norm is also known as the Chebyshev radius.

$$
\begin{aligned}
&\min_{\mathbf{x}} \quad f^1(\mathbf{x}) && (1.8)\\
&\text{such that } \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n\\
&\qquad\downarrow\\
&\min_{\mathbf{x}} \quad f^2(\mathbf{x})\\
&\text{such that } \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n\\
&\qquad f^1(\mathbf{x}) \leq f^1(\mathbf{x}_1^*)\\
&\qquad\vdots\\
&\min_{\mathbf{x}} \quad f^m(\mathbf{x})\\
&\text{such that } \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n\\
&\qquad f^i(\mathbf{x}) \leq f^i(\mathbf{x}_i^*) \quad \forall i \in \{1, 2, ..., m-1\}
\end{aligned}
$$

Lexicographic methods get their name from the idea of lexicographic order, this is a generalization of an alphabetical ordering scheme. In the context of an optimization problem, this type of method might be used when different objectives cannot easily be compared quantitatively, and/or there is a clear order of objective importance [12]. Anytime designer preferences are such that $f^1(\mathbf{x}_1) < f^1(\mathbf{x}_2)$ implies $\mathbf{x}_1$ is preferable to $\mathbf{x}_2$ (independent of $f^2(\mathbf{x}), ..., f^n(\mathbf{x})$), a lexicographic optimization approach is suited to capture these preferences.

1.2.5 Goal Programming

The philosophy of goal programming is to find solutions which are "satisficing" rather than necessarily optimal.[4] Goal programming seeks to attain user-defined levels of each objective as closely as possible rather than simply pursuing designs that are necessarily optimal. Goal programming can be a useful option if a designer is able to specify desired levels of each objective a priori. In fact, the approach has been successfully used in many real-world problems including the design of an antenna system for the Saturn launch

[4]The term "satisficing" is a combination of the words "satisfying" and "sufficient." The word was first used by Herbert A. Simon (1916-2001), a professor at Carnegie Mellon who won the Nobel Prize in economics for his work in decision-making processes [13]

vehicle [4].

In goal programming, each objective in the MOO problem is assigned some ideal value, termed an "aspiration."[5] A "goal" technically refers to an aspiration/objective pair. The scalar function that is minimized in goal programming is a function of all deviations from this goal. The general framework of goal programming allows designers to treat positive and negative deviations differently in their achievement function. We can formulate our general MOO problem as a goal programming problem as follows:

$$
\begin{aligned}
& \min_{\mathbf{x},\mathbf{p},\mathbf{r}} \quad A(\mathbf{p},\mathbf{r}) && (1.9)\\
\text{such that} \quad & \mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^n, \ \mathbf{p},\mathbf{r} \in \mathbb{R}^n \\
& f^i(\mathbf{x}) - r_i + p_i = a_i \quad \forall i \in \{1,...,m\} \\
& r_i p_i = 0 \quad \forall i \in \{1,...,m\} \\
& r_i, p_i \geq 0 \quad \forall i \in \{1,...,m\} \\
\text{where:} \quad & \mathbf{p} \triangleq \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}, \mathbf{r} \triangleq \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}
\end{aligned}
$$

In the above, the p_i and r_i are the positive and negative deviations of objective i from aspiration i. The objective function in this framework, $A(\mathbf{p},\mathbf{r})$ is termed the achievement function [14]. The goal programming framework allows users to treat positive and negative deviations in a given objective differently. Suppose profit were one of a designer's objectives for example, let us say f_2. If profit level were above the aspiration level, c_2, there would have to be a nonzero value for the positive deviation variable p_2. But because excess profit is not a negative outcome, a designer might decide not to let p_2 appear in the achievement function. This type of behavior is not possible with a distance function.[6]

One useful application of goal programming relates to multidisciplinary design optimization (MDO). In general, MDO approaches seek to break down

[5]The set aspirations over all objective functions is very similar to the "target" point in the distance method already described.

[6]It is actually not strictly necessary to include the constraint $r_i p_i = 0 \quad \forall i \in \{1,...,m\}$ (a form of constraint called a "complementarity constraint"). This is because the constraint will be satisfied by default at any optimum as long as the positive and/or negative deviation variables appear in the achievement function.

large multidisciplinary engineering problems in to constituent subsystems. Approaches seek to optimize each subsystem, while enforcing compatibility constraints between them to ensure overall system feasibility [15]. MDO frameworks attempt to solve this optimization and coordination problem. Simpson et al used goal programming to allow multiobjective design within a given subsystem in an MDO problem [16].

1.2.6 Genetic Algorithms in Multiobjective Problems

Genetic algorithms (GAs) are a class of gradient-free optimization methods that are inspired by the process of biological evolution. Before they were applied to solve multiobjective problems, GAs were used to solve problem formulations with a single objective. There is a large body of research devoted to genetic algorithms or so-called evolutionary algorithms, beginning with seminal works of John Holland of the University of Michigan [17].[7] Here I provide a concise overview of some of the terminology and strategy of genetic algorithms in general, and then discuss applications of genetic algorithms to multiobjective problems. This is not meant to be an exhaustive overview (which would be unrealistic), but rather to familiarize the reader with the application of genetic algorithms to multiobjective problems. The research I have performed in multiobjective optimization makes use of a multiobjective genetic algorithm (MOGA), so of all the methods already described, this is the most relevant to the research work presented.[8]

The overall flow of any genetic algorithm follows figure 1.3[9] The idea of the genetic algorithm is to allow the fittest individuals in the population to pass on their most desirable qualities to future generations. The process is analogous to the evolution in biology. Initially, a population is randomly generated where each member of this population corresponds to a single design vector. In a single objective problem, each member of the population can be assigned a score based on the objective function value attained by

[7]Professor David E. Goldberg, a fellow Illini, is another well-known researcher in the field of genetic algorithms. He was an advisee of Doctor Holland at the University of Michigan, and contributed many highly cited works in the field including [18].

[8]The interested reader is referred to Kalyanmoy Deb's book *Multiobjective Optimization using Evolutionary Algorithms* for a more comprehensive review of genetic algorithms applied to multiobjective design optimization [19].

[9]This flowchart is taken from [19], with some minor modifications.

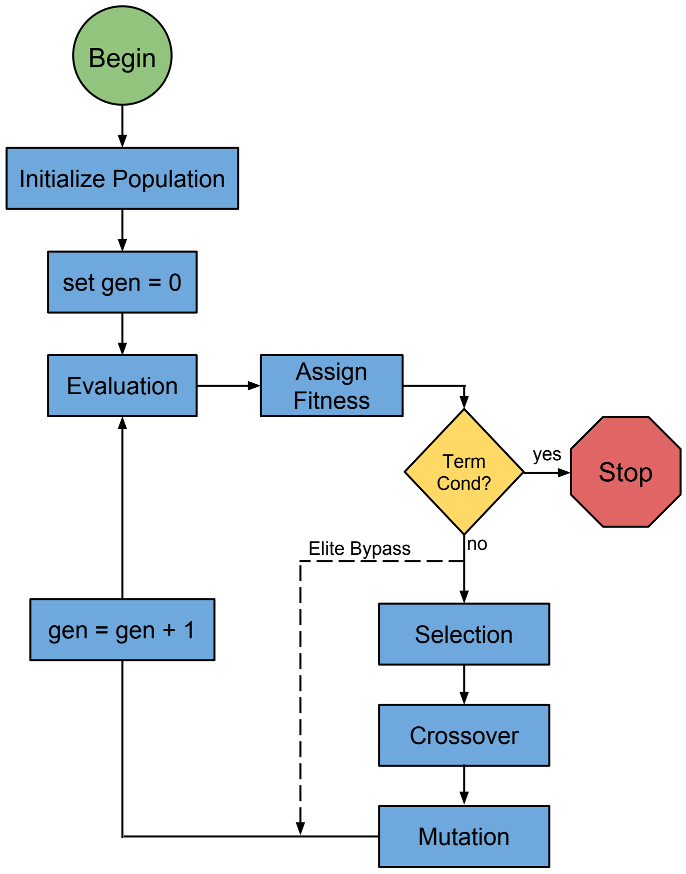


Figure 1.3: Genetic algorithm flow chart (adapted from [19])

that design. A population member's objective score then maps to a "fitness" value. Based on their fitness values, and a stochastic selection process, individuals from the current generation are chosen to be part of the mating pool. This mating pool undergoes two operations, crossover and mutation. During crossover, designs in the mating pool make "children" designs, which result from forming a new population member from the design vectors of two members of the mating pool. After crossover, mutation randomly varies some or all of the elements in a population's design vectors. The dotted line in figure 1.3, is an optional measure used to preserve the most elite members of a population. Some fraction of the current population can be passed unchanged into the next generation, as well as being allowed to be part of the mating pool. This "elite bypass" ensures that the fittest members in each subsequent population cannot degrade through random changes of crossover and mutation.

Even a cursory read of the above paragraph should leave many questions in the reader's mind. How are scores in an objective function mapped to fitness values? Which stochastic process selects members of a mating pool? How are parent's paired? What is the magnitude of variation in a design vector which results from mutation? What fraction of a population should be chosen as elite, and thus guaranteed members of the next generation population? All these questions are the subjects of numerous research works that have continued for decades since the first application of genetic algorithms in the 1960s and 1970s by John Holland's group at the University of Michigan.

Genetic algorithms have been generalized to deal with multiobjective problems by taking advantage of the concept of non-dominance [20].[10] Ranking of design i is accomplished by counting the number of designs that dominate it. In this way, all non-dominated solutions are assigned the highest rank, and thus emphasized in future generations. With this approach, the same methods used in single objective genetic algorithms can be applied to multiobjective problems.

Application of genetic algorithms to multiobjective problems has one considerable advantage over more conventional approaches to multiobjective optimization; after a single successful application of a genetic algorithm, a designer is presented with a final *population* of designs, rather than just a single design. With an appropriate set of genetic algorithm parameters, the non-dominated solutions on the final population give an indicator of the problem's Pareto set, allowing genetic algorithsm to be a useful generating method for multobjective problems.

Focus now shifts to a case study. In the next chapter, I use a multiobjective genetic algorithm to aid in the design of a reconfigurable Delta robot.

[10]To clarify, a point in the attainable set is considered non-dominated if there does not exist any other point in the attainable set that dominate it, i.e. there does not exist any other point in the attainable set that performs better in objective and no worse in all others.

CHAPTER 2

APPLIED MULTIOBJECTIVE OPTIMIZATION: RECONFIGURABLE SYSTEM DESIGN

2.1 Introduction

Having given a broad overview of the subject of multiobjective optimization, the discussion turns now to applied work in multiobjective optimization—specifically, in the area of reconfigurable system design. First, motivation for research in reconfigurable systems is presented, followed by a discussion of a new design method for multi-ability reconfigurable systems and a presentation of selected research results.

2.2 Motivation

Reconfigurable systems have been defined in literature as "those systems that can reversibly achieve distinct configurations (or states), through alteration of system form or function, in order to achieve a desired outcome within acceptable reconfiguration time and cost [21]." (Systems that are able to change their states or configurations have also been called "adaptable" [22], or "flexible" [1,23]. We will continue using the term "reconfigurable" for consistency, but note in passing that the use of the adjective "flexible" to describe reconfigurable systems could cause confusion, especially when designing systems with elastic compliance, i.e. mechanical flexibility.)

Many practical applications benefit from the implementation of reconfigurability in design. A prime example is manufacturing. According to Koren, high responsiveness is among the three principal goals of modern manufacturing systems [24]. In 1996, the state of Michigan, the National Science Foundation, and several major manufacturing companies provided $47 million in seed money to found the Engineering Research Center for Reconfigurable Manufacturing Systems at the University of Michigan. This funding resulted in over 1700 papers published about reconfigurable manufacturing systems by 2010. Reconfigurable manufacturing systems offer firms the ability to change manufacturing capacity and functionality quickly, allowing fast response to unpredictable changes either externally (e.g. in market demand) or internally (e.g. in necessary machine maintenance) [25].

Reconfigurability also finds application in aircraft design. Morphable airfoils are a reconfigurable system with the potential to increase the efficiency and maneuverability. Because a single aircraft may assume many distinct operating states over a typical mission (loitering, climbing, cruising, etc.), and because a fixed (non-reconfigurable) system is not capable of providing optimal performance over multiple different operating states, the system is a good candidate for reconfigurability. The performance benefits of changeable wing shape have been explored at least since the beginning of manned flight (the Wright Flyer in 1903), and have seen continued application in many other commercial aircraft to date (e.g., the Grumman F-14 Tomcat, and the Rockwell B-1 Lancer) [26].

Another application of reconfigurability is modular robotics. The goal of the work in this domain is to develop a versatile, cheap, and controllable robot made up of self-similar modules. PolyBot (figure 2.1) is one example of research aimed at achieving this goal. Developed at the Xerox Palo Alto Research Center (PARC), Polybot is a composed of many robotic units that function as actuated hinges in the assembled design. By altering the connections between units, the robot reconfigures itself into different structures, each capable of a different gait. Attainable structures include a four-legged walking robot, a snake-like chain capable of "slithering," and a circular structure capable of moving like the tread on a tank [27].

These examples only scratch the surface of the applications of reconfigurability. Many more systems in industry and academia serve as excellent examples of exploitation of reconfigurability. The interested reader is referred to [21] for a more exhaustive list.

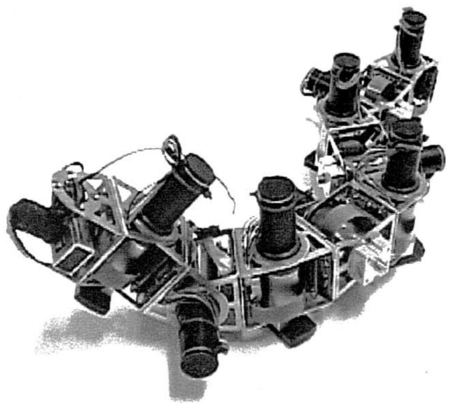


Figure 2.1: Polybot robot developed at Xerox Palo Alto Research Center [27]

2.3 Literature Review and Background

This section is divided into two parts. First, we focus on a review of studies that target design for reconfigurability, adaptability, or flexibility. Second, we review the basic problem structure in product family design and the insight it offers into the reconfigurability problem.

2.3.1 Classification of Reconfigurable Systems

In the design research community, reconfigurable system design has received considerable attention in recent years. Common issues that are discussed in the literature include: (1) generation/selection of reconfigurable design concepts, (2) estimation of the cost of reconfigurability, (3) classifications of reconfigurable systems, and (4) selection of reconfigurable variables.

Research in reconfigurable design concept generation and selection aims to provide quantitative methods to discount inferior design concepts as early as possible. Mattson and Messac distinguish clearly between the terms "concept generation" and "alternative generation"—the prior referring to generation of topologically different designs, and the latter referring to different embodiment within a particular system architecture. In the same work, Mattson et al. introduced the idea of s-Pareto optimality, a set-level parallel of Pareto optimality. The authors show that a design concept can be represented by an attainable Pareto set in the objective space. When the Pareto sets of multiple concepts are combined in the performance space, the non-dominated points on this combined surface form the s-Pareto frontier [28]. Literman, Cormier, and Lewis developed an iterative framework specifically to generate promising design concepts for reconfigurable systems [29].

Classifying the types reconfigurable systems and the underlying motivations for reconfigurability helps to break down design problems into more manageable parts. For example, the designer's approach in a *modularly* reconfigurable system being designed for *robustness* should be very different from the approach used in a *continuously* reconfigurable system being designed for *adaptability.* To sharpen the meanings associated with terms like "robustness", "adaptability", and "modular reconfigurability", there was a need for clear classifications. In this vein, Olewnik et al. introduced a hierarchical classification of reconfigurable systems [1], which is illustrated in figure 2.2. The most general class of systems was termed "open", which is a generalized term used originally by Simpson to describe "systems of industrial products, services, and/or processes that are readily adaptable to changes in their environment..." [30]. The hierarchy of Olewnik et al. differentiates between "flexible" and "modular" systems; system adaptability versus system robustness; and "passive" versus "active" adaptability. Definitions from the paper are included here for completeness. They are taken directly from [1].

Flexible systems: Systems designed to maintain a high level of performance through real-time adaptations in their configuration and/or through robust parameter settings when operation conditions or requirements change in a predictable or unpredictable way. This definition implies that flexibility can be obtained through two modes: adaptability and robustness.

Adaptability: Mode of achieving flexible systems where system parameters (design variables) that can be changed and their range of change are identified to enchance performance of the system in *predictable* changes in the operating environment; they can be changed when the system is not in use (passive) or in real time (active)

Robustness: Mode of achieving flexible systems where system parameters (design variables) are set constant to minimize the effect of *unpredictable* changes in the operating environment on the performance of the system without eliminating the cause of the changes themselves.

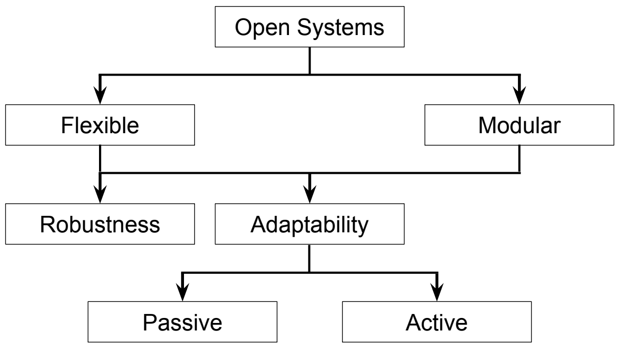


Figure 2.2: Heirarchy of open systems proposed by Olewnik et al. [1]

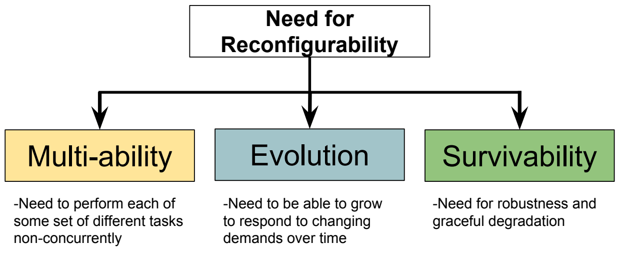

Need for Reconfigurability
Multi-ability
Evolution
Survivability
-Need to perform each of some set of different tasks non-concurrently
-Need to be able to grow to respond to changing demands over time
-Need for robustness and graceful degradation

Figure 2.3: The underlying motivations for reconfigurable design [2]

Another important classification was made by Siddiqi, De Weck, and Iagnemma. They identified three system requirements that motivate incorporation of reconfigurability [2]: (1) multiability, (the ability of a single system to perform different tasks non-concurrently) (2) evolvability, (the ability of a system to adapt to known or unknown changes in its future operating environment) and (3) survivability (the ability of a system to continue functioning despite failure in one or more components). Figure 2.3 diagrams these ideas.

Total cost of reconfigurability is an essential part of the decision of whether or not to incorporate it in a design. Further, the relative cost of reconfigurable system components often drives the reconfigurable design variable selection problem. Patterson, Pate, and German detailed many of the benefits associated with designing a modularly reconfigurable family of UAVs [31]. The chief motivating factors for this consideration were both fleet evolvability (ability to adapt to known and unknown future field conditions) and cost (because a modular fleet might also be less expensive to maintain). Ferguson and Lewis considered the performance cost associated with the potential added mass required to incorporate reconfigurability [32]. Ferguson, Kasprzak, and

Lewis went on to use this performance cost to help decide which system variables should be made reconfigurable in the design of a family of formula race cars [33]. Khire and Messac identified the similarity between the variable selection problem in reconfigurable system design, and the platform selection problem in product family design, exploiting it in their solution to the variable selection problem [22].

2.3.2 Relationship Between Reconfigurable Systems and Product Families

At this point, the parallels between product family design and reconfigurable system design may have already occurred to the reader. This connection has been made by more than a few authors in literature [22,23,31,33], especially those who contribute to both reconfigurable design and product family design communities. We outline the premise of product family design below and then describe the similarities it bears to the reconfigurable system design problem. The interested reader is referred to [34] for a more in-depth review of product family design.

At a high level, product family design research aims to find design approaches that maximize profit over all the market segments they serve. This single objective drives firms to simultaneously maximize commonality over their product portfolio (because it costs less to produce a product portfolio with more commonality), while meeting necessary performance metrics in all their target market segments. This dichotomy is commonly referred to in literature as the "commonality vs. performance" tradeoff. Firms respond to this dichotomy by designing core components and technologies that can be shared between many of their products. Black & Decker® power tools are a well-known example of a product line leveraging common core technologies. The set of elements that are shared among multiple products is called a product platform. Collectively, the products that share a platform are known as a product family.

Product family design approaches are classified broadly into two groups: (1) module-based and (2) scale-based. In module-based product family design, product platforms are functional modules shared over the products of a firm. For example, the chassis and suspension of a car company's luxury sedan might be identical to those of its lower-end models. In this case, the chassis and suspension module would be part of the product platform. In a scale-based product family design, elements of a product platform are technologies or components which designers can “stretch” or “shrink” with minimal redesign.

Product family design requires designers to consider questions that extend beyond single product design in order to remain competitive in all market segments. How many product families should a firm offer? What components should make up each product platform? How can designers optimize the design of a single platform that is used in multiple end products? Justifying design decisions requires information about customer preferences, manufacturing costs, and performance capabilities. Research work in this field has sought to provide systematic approaches to simplify the many decisions necessary in product family design.

The “commonality vs. performance” tradeoff in product family design is analogous to the “reconfigurability vs. performance” tradeoff in reconfigurable system design. Increasing reconfigurable capability supports better performance for each system function, but results in greater system cost. Reducing reconfigurable capability results in a simpler, less-expensive system, but sacrifices the ability to tailor system configuration to optimize performance for each system function. Reducing reconfigurable capability increases commonality between system configurations (modes of operation), similar to increasing commonality in a product family at the cost of reduced performance across market segments.

In this work, we develop the reconfigurable design variable selection problem for a particular class of reconfigurable systems: continuously reconfigurable systems designed for multiability through offline transformation. We show that the reconfigurable system designed for multiability bears many similarities to a formulation originating from product family design, and adapt this formulation to our problem. Further, we show that for this specific class of reconfigurable systems, the designer does not need to compare the attainable Pareto fronts of different reconfigurable candidates, and can instead formulate a larger multiobjective problem whose Pareto optimal points will each encode fully reconfigurable designs.

Consider the design of a continuously reconfigurable system for multiability. We use the term “continuously reconfigurable” to mean that all reconfigurable variables can vary continuously along a prescribed range. Suppose that we seek to improve the performance of the system in m discrete tasks. Suppose the system architecture has been specified a priori, and that our goal is to determine optimal ranges for reconfigurable design variables. (It is worth noting that an “optimal” reconfigurable range may be zero.) Solving this variable selection problem means selecting not only which variables are reconfigurable, but also to what extent. Let the number of design variables necessary to specify a fixed (i.e. non-reconfigurable) design be n and the number of tasks required of the reconfigurable system be m. Let $f^i(\cdot)$ be the negative of the utility function for task i and the scalar function $c(\cdot)$ represent the cost of a reconfigurable system. We can formulate the following multiobjective optimization problem:

$$\begin{aligned} \min_{\overline{\mathbf{x}}} \quad & \{f^1(\mathbf{x}^1), ..., f^m(\mathbf{x}^m), c(\overline{\mathbf{x}})\} \\ \text{subject to} \quad & \mathbf{g}_i(\mathbf{x}^i) \le 0 \quad \forall i, \quad 1 \le i \le m \\ & \mathbf{h}_i(\mathbf{x}^i) = 0 \quad \forall i, \quad 1 \le i \le m \\ \text{where:} \quad & \\ & \overline{\mathbf{x}} := \begin{bmatrix} \mathbf{x}^1 \\ \vdots \\ \mathbf{x}^m \end{bmatrix}, \quad \mathbf{x}^i := \begin{bmatrix} x^i_1 \\ \vdots \\ x^i_n \end{bmatrix}. \end{aligned} \tag{2.1}$$

$\overline{\mathbf{x}}$ is a vector of design variables composed of individual design vectors, where $\mathbf{x}^i$ is a design vector that describes the ith system task. Thus, the design vector for the full reconfigurable system design problem has length mn, and specifies the values of all n (possibly reconfigurable) design variables in all m tasks. The inequality and equality design constraints (if present) are represented by $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$, respectively. If reconfigurability cost nothing, the solution to this multiobjective problem would simply be a concatenation of the solutions to each single-objective problem. When realistic cost function $c(\cdot)$ is used, however, the tradeoff between reconfigurability and cost presents itself, forcing the designer to choose between cost and performance.

Compare the above optimization formulation with the following product family design formulation, adapted from Fellini’s product platform selection problem [35].

$$\max_{\eta, \overline{\mathbf{X}}=[\mathbf{x}^1,...,\mathbf{x}^m]} \{f^1(\mathbf{x}^1), ..., f^m(\mathbf{x}^m), \sum_{(i,j)pq} \eta_{ij}^{pq}\}$$
$$\forall p, q \in \mathcal{P}, (i,j) \in \mathcal{S}^{pq}, p < q$$

$$\begin{aligned} \text{subject to} \quad & \mathbf{g}^p(\mathbf{x}^p) \leq \mathbf{0} \\ & \mathbf{h}^p(\mathbf{x}^p) = \mathbf{0} \\ & \eta_{ij}^{pq}(x_i^p - x_j^q) = 0 \\ & \eta_{ij}^{pq} \in \{0,1\} \end{aligned} \tag{2.2}$$

where:

f^p	performance function for product p
$\mathbf{x}^p$	design vector for product p
$\mathcal{P}$	set of indices, each referring to a product
m	number of products and the cardinality of $\mathcal{P}$
S^{pq}	index pairs of elements that are candidates for being shared between two products p and q
$\mathbf{h}^p$	equality constraints in product p
$\mathbf{g}^p$	inequality constraints in product p
η_{ij}^{pq}	binary variable indicating if components i and j are shared between products p and q

Notice that in both the reconfigurable system optimization formulation and the product platform design formulation, the commonality vs. cost dichotomy arises. At an abstract level the two problems are nearly identical. The implementation of the optimization formulation given in Eqn. 2.1 will be explored further in the next section.

2.4 Variable Selection Problem For Continuously Reconfigurable Systems For Multiability

Because so many systems fit under the umbrella term "reconfigurable systems," it can be difficult to arrive at specific design strategy capable of treating all of them. The nature of the reconfigurable design problem depends on the goal sought and the method of reconfigurability. For example, systems that are modularly reconfigurable require a different approach than those that are continuously reconfigurable. This work treats the design of a particular type of reconfigurable system: "offline continously reconfigurable," for a particular purpose: "multiability." Further, we assume that the system architecture has already been selected, and that we are looking to solve the variable selection problem. That is, we are looking to decide which variables to make reconfigurable and over what ranges. Having specified this class of reconfigurable systems as the scope of our study, we can proceed with a more in-depth analysis.

We can now discuss the design variables of the reconfigurable system and, because we have targeted multiability as our goal, we can talk about set of tasks that we are interested in having the system perform. Let m be the number of tasks the reconfigurable system is to perform. The reconfigurable system, therefore, will have at most m different configurations, i.e. at most m distinct n-dimensional design vectors, each specifying a nonreconfigurable (static) design.

Considering m different tasks for the system, we should be able to establish utility functions for each task, which depend on the values of the static design variables in each configuration. (In the case of a manufacturing system being designed for multiple machining operations, these utility functions might correspond to the times required to complete reference tasks in each configuration.) Call the set of m utility (objective) functions $f^1(\cdot)$, $f^2(\cdot)$, etc. such that $f^i(\cdot)$ corresponds to task i.

To better describe this variant of the reconfigurable system design problem, we introduce a simple multiobjective optimization example problem, and then recast it as a simultaneous variable selection and optimization problem for reconfigurable system design. Consider the following simple multiobjective optimization problem:[1]

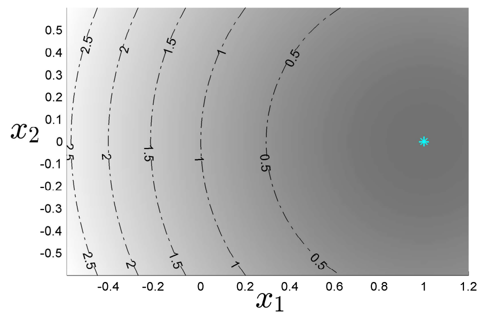
[1]The reader might recognize this problem from Chapter 1.

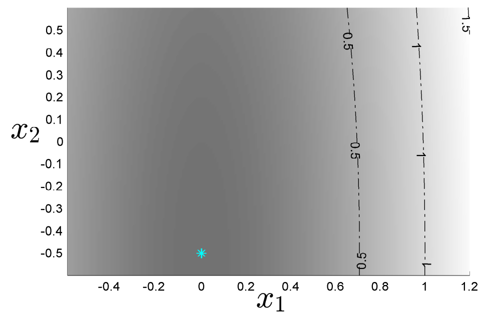
$$\begin{aligned} \min_{\mathbf{x}=[x_1,x_2]^T} \quad & \{f^1(\mathbf{x}), f^2(\mathbf{x})\} \\ \text{where:} \quad & f^1(\mathbf{x}) = \left(\frac{x_1-1}{1^2}\right)^2 + (x_2)^2 \\ & f^2(\mathbf{x}) = (x_1)^2 + \left(\frac{x_2+0.5}{2^2}\right)^2 \end{aligned} \tag{2.3}$$

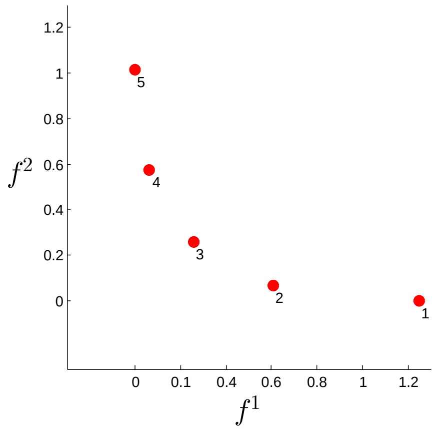
Because this multiobjective optimization problem happens to have 2 design variables and 2 objective functions, we can plot points in the design space, along with their corresponding images in the function space. These plots are shown in figure 2.4.

Selecting ranges of reconfigurability for design variables usually means attempting to “value” subsets of the Pareto set in the performance space, and choose optimum ranges for each reconfigurable variable such that this value is maximized. Consider figures 2.4c and 2.4d. The points shown in these figures lie in the Pareto set of the nonreconfigurable (static) design problem—each point corresponding to a fixed design. In this example, the variable selection problem amounts to choosing two ranges, (one each for x_1 and x_2) which capture the “best” subset of the Pareto set in the static design problem. For example, a designer might choose bounds for reconfigurable variables in the design space that capture points 2, 3, and 4 in figure 2.4c. Valuing subsets of this Pareto set can be difficult and requires prior determination of the Pareto frontier before decisions of reconfigurability can be made. In the continuously reconfigurable design problem for multiability, we are only interested in the system's individual performance in each of the m distinct tasks. Since each task has its own objective function, the “value” of a subset of the nonreconfigurable Pareto front is captured only by the m points on the subset of the Pareto front that individually minimize each of the objective functions $f^i(\cdot)$ individually. (i.e., the points that minimize the $f^i(\cdot)$ value achievable by the given reconfigurable system design $\forall i = 1, 2, \ldots m$).

Provided that cost can be quantified as a function of the extent of reconfigurability, the variable selection problem can be recast into the following simple form, whose performance space is only one dimension higher than the performance space of the fixed problem.


(a) f^1 objective surface with minimum value shown at starred point


(b) f^2 objective surface with minimum value shown at starred point


(c) 5 points along the Pareto frontier

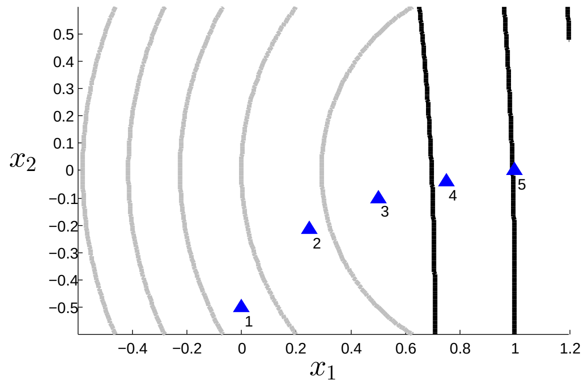

(d) Preimage of the 5 points shown in figure 2.4c

Figure 2.4: Multiobjective Design Problem With 2 Objectives and 2 Design Variables

Consider the following reformulation:

$$\min_{\mathbf{x^1},\mathbf{x^2}} \quad \{f^1(\mathbf{x^1}), f^2(\mathbf{x^2}), c(\mathbf{x^1},\mathbf{x^2})\} \tag{2.4}$$

$$\text{where:} \quad f^1(\mathbf{x}) = \left(\frac{x_1 - 1}{1^2}\right)^2 + (x_2)^2$$

$$f^2(\mathbf{x}) = (x_1)^2 + \left(\frac{x_2 + 0.5}{2^2}\right)^2$$

In this formulation, the two vectors $\mathbf{x^1}$ and $\mathbf{x^2}$ specify a reconfigurable design by specifying the system configurations for each task. Based on these two (in general m) vectors, the ranges of reconfigurability of each design variable can be determined. The optimal design vector for the reconfigurable design problem contains more information than just the required ranges for each design variable. It also specifies an optimal system configurations for each task.

Note the changes from Eqn. (2.3) to Eqn. (2.4). In Eqn. (2.4) we add one design vector to the original problem and a new function $c(\cdot)$ that quantifies the cost of reconfigurability. The Pareto set generated by this formulation expresses the tradeoff between reconfigurable system performance in individual tasks and the cost of increasing the level of reconfigurability. The goal of the designer is to select a point along this Pareto set, thereby selecting a design which is non-dominated in the context of the multiability problem.

The performance space of this multiability problem consists of three dimensions ($m + 1$ in general), one for cost and one for each of the m performance functions. In the case of the multiability problem, it is very important to note that the designer is only concerned with the extreme points on any candidate subset of the fixed system's Pareto frontier. This simplification of the design problem stems from the assumption that we are only concerned with m separate tasks individually.

Consider a given point in the Pareto front of Eqn. (2.3) and how it compares to a point on the Pareto front in Eqn. (2.4) in both the objective and design space. Points in the Eqn. (2.3) Pareto set fall along a continuous curve that goes from point 1 to point 5 in Figs. 2.4c and 2.4d. The points in the design space (figure 2.4d) and objective space (figure 2.4c) each represent a given design with fixed design variable values. In contrast, points in the Eqn. (2.4) Pareto set each correspond to a reconfigurable system on a larger Pareto front. In the objective space, the Pareto set of this second formulation must include the Pareto set of the original nonreconfigurable formulation; because cost of all points on the first Pareto set is zero, none can be dominated by a reconfigurable system design on the expanded Pareto set. This expanded multiobjective Pareto set provides valuable information during early stage design, making the price of reconfigurability and its tradeoff clear to the designer. This Pareto set surface in the objective space is depicted in figure 2.5.

A few features are important to note. As was already mentioned, the lowest cost cross-section of the surface corresponds to the Pareto set of the multiobjective optimization of the fixed system. Secondly, the utopia point

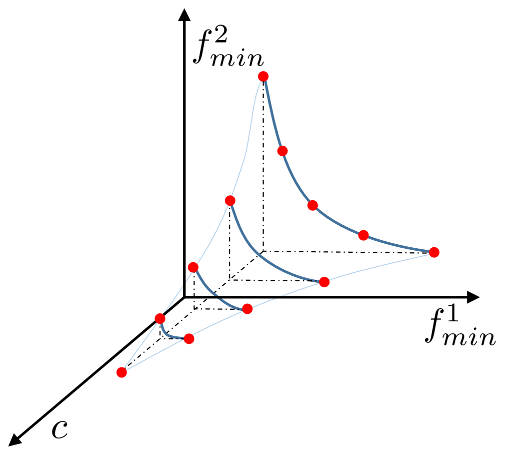


Figure 2.5: Expanded Pareto surface for a continuously reconfigurable system designed for multiability

of the lowest cost Pareto set (in fact the utopia point of Pareto sets at all cost levels) is the only coordinate of the Pareto surface at the highest cost level. This must be true, since allowing the system to reconfigure completely should allow it to achieve the performance extremes of the fixed design. Finally, consider that, as cost increases, the Pareto set cross-sections must perform at least as well as cross sections at lower cost levels.

The important point here is that for continuously reconfigurable systems designed for multiability, this surface can be a useful early stage design tool that demonstrates the cost of reconfigurability from a fixed design to a fully reconfigurable one. Rather than calculate the full Pareto sets of each reconfigurable system and compare them, (in the case of multiability) a designer can focus on only m important points on each Pareto set (i.e. those which minimize performance functions individually). When plotted against the cost objective function (e.g., figure 2.5), the designer is presented with a Pareto front where each point refers to a Pareto optimal reconfigurable system. Aside from the additional cost axis, Note the change in axis labels from f^1 and f^2 in the non-reconfigurable problem to f^1_{min} and f^2_{min} in the reconfigurable one.

In the next section, we attempt to find points along this surface in the case of a parallel machine tool being designed for multiability. This case study will serve to help demonstrate the usefulness of the expanded Pareto surface introduced in this paper.

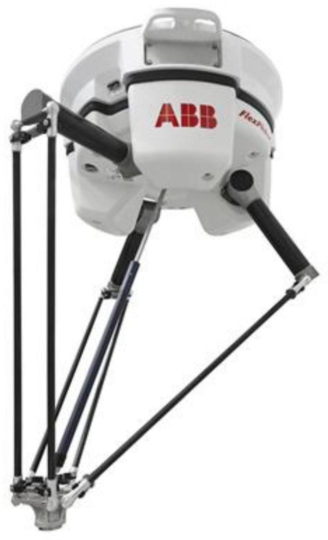


Figure 2.6: The ABB Flexpicker®

2.5 Case Study: Reconfigurable Delta Robot Design

2.5.1 Overview

The Delta robot is a type of parallel robot which is customarily used for high-speed pick-and-place operations. Kinematically, the Delta is specially designed so that the orientation of its end effector is constant, i.e. all end effector motions are pure translations. Originally developed by Clavel and his research team at EFPL (École Polytechnique Fédérale de Lausanne) in the early 1980s [36], the robot is now used at an industrial level for high-speed pick-and-place operations. Examples of commercial and industrial application of the robot include the FlexPicker® (figure 2.6) and the SurgiScope®.

2.5.2 Framing The Design Problem

The current case study treats the design of a continuously reconfigurable delta robot for two tasks. The first task is a high speed pick-and-place operation of payloads with negligible size. The second task is a milling operation, requiring system rigidity rather than speed. We compare designs by evaluating stiffness and speed characteristics of each point in the build area along

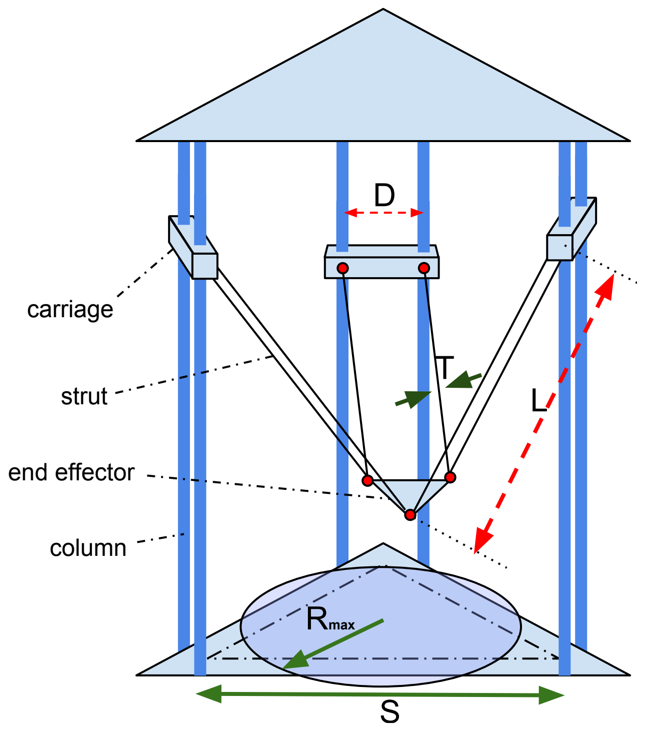


Figure 2.7: Design architecture used in case study. The two design variables are marked with dashed lines.

a polar grid. Pointwise performance is then used to calculate performance functions for stiffness and speed that apply to the whole design.

The goal of the case study is to determine reconfigurable designs which make up the reconfigurable system Pareto set described in Section 2.4. The study will illustrate one method for finding this Pareto frontier, as well as its value as an early stage design tool.

The proposed design architecture is shown in figure 2.7. All joints in the design are spherical. A summary of all design variables and parameters in the figure is included in Table 2.1. Most variables listed in Table 2.1 are self-explanatory. We would however like to clarify the meaning of the variables S and R_{max}.

The '**S**' Parameter

Solving the forward kinematics problem in this Delta architecture, i.e., solving for the end effector location given the location of the carriages, amounts to finding the intersection of three spheres as they move in the z direction.

Table 2.1: Parameters and design variables used In reconfigurability case study

Variable	**Type**	**Units**	**Nominal Value or Range**
D	design var.	m	$0.05 - 0.35$
L	design var.	m	$(0.6 + R_{max}) - 3.5$
S	parameter	m	1
R_{max}	parameter	m	$0.7\frac{S}{\sqrt{(3)}}$
T	parameter	m	0.05
E	parameter	Pa	2×10^9

These spheres are not centered exactly on the carriages as one might expect, however. Instead, their center locations are offset toward the end effector by $\frac{D}{2\sqrt{3}}$. This is to account for the size of the end effector itself. The distance S, which is held constant in this case study, is the distance between these kinematic spheres as measured in the xy plane. Holding this distance constant allows the kinematics of the Delta to be treated independently of the separation distance D. In this reconfigurable problem it is assumed that reconfigurability in the D variable is accompanied by changes in column separation necessary to preserve the distance S.

The '$\mathbf{R_{max}}$' Parameter

The R_{max} parameter measures radius of the build area of interest. This value is necessary to specify a priori so that Delta robot designs can be compared using local stiffness and speed metrics at a standard set of points in the build envelope. Calculating rigidity metrics over the reachable envelopes of each design and comparing them would not be an apples-to-apples comparison. Thus, there is a need for a standard build area over which we can compare all designs fairly. For the purposes of this case study we happened to choose the radius of our desired build envelope as $R_{max} = 0.7\frac{S}{\sqrt{3}}$.

2.5.3 Performance Functions

In order to frame the reconfigurable variable selection problem as was done in Section 2.4, we need to provide performance functions for each task. In the current problem, we need two performance functions capable of quantifying the speed and rigidity of a given delta robot geometry, respectively.

Rigidity Performance Function

Our approach in defining a stiffness performance function is an adaptation of that used by Courteille, Deblaise, and Maurine [37]. In their work, Courteille et al. design a similar parallel robot for rigidity. Courtielle et al. use the alternative eigenvalue problem proposed by Lipkin and Patterson [38] to calculate the principle angular stiffnesses, and principle linear compliance directions in a Delta-like robot. We now provide a general overview of this approach and discuss its implementation in our system. For more details on the theory behind the method, the interested reader is referred to the original articles [37, 38].

The performance function used in this design problem comes from a linear finite element model of the Delta robot. For a set of n regularly-spaced positions within the desired build envelope, we measure the stiffness of the robot end effector as if the Delta were a static structure comprised of beams. This gives us the ability to quantify the stiffness of the robot at any position by using the direct stiffness method. The result is a 6×6 stiffness matrix for each point where we perform the calculation. In this static model of our Delta, we assume that compliance occurring in the columns of the robot are negligible. A diagram of our node and element numbering scheme is included in figure 2.8 for reference. (Note that the nodes have circular labels, and the elements have triangular labels.)

Lipkin and Patterson proposed the "alternative eigenproblem" to decompose the 6×6 stiffness matrix at each point into principle rotational stiffnesses and linear compliances. The alternative eigenproblem is stated as follows:

$$c_{f,i}\mathbf{\Gamma}\overline{F}_i = \mathbf{C}\overline{F}_i \tag{2.5}$$

$$k_{\gamma,i}\mathbf{\Omega}\overline{x}_i = \mathbf{K}\overline{x}_i \tag{2.6}$$

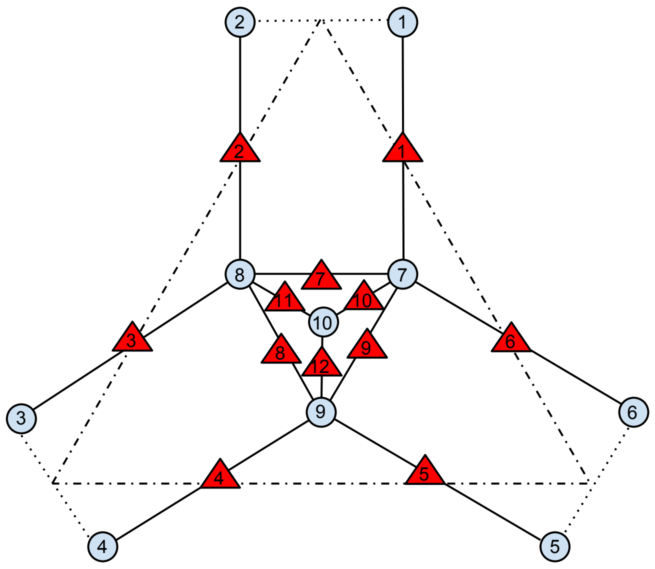


Figure 2.8: Node and element numbering used in Delta stiffness calculation (Nodes are labeled with triangles while elements are labeled with circles)

where:

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \quad \mathbf{\Omega} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$$

In the alternative eigenproblem $\overline{F}$ is a generalized force vector made up of three translational forces and three torques, termed a wrench. Similarly, $\overline{x}$ is a generalized displacement vector made up of three linear displacements and three rotational displacements, termed a twist. Generalized force vectors that satisfy Eqn. (2.5) produce pure translations in the end effector parallel to the applied translational force and are termed eigenwrenches. Similarly, generalized displacement vectors that satisfy Eqn. (2.6), result in purely rotational reaction force parallel to the applied displacement and are termed eigentwists.

By comparing the eigenvalues of the alternative eigenproblem, a designer is capable of assessing the level of translational and rotational stiffness in three principal directions. These eigenvalues can be used to form local stiffness metrics which indicate the magnitude and uniformity of stiffness at a given location in the build envelope.

While Lipkin and Patterson's approach can be applied to each point in the build envelope, Courteille et al. suggest global metrics, which combine stiffness information from a collection of n points from throughout the build envelope. They combine local stiffness/compliance matrices to form global stiffness/compliance matrices, and then use a singular value decomposition to determine global indices. The global stiffness and compliance matrices are calculated as follows:

$$\widetilde{\mathbf{K}}^G = \begin{bmatrix} \widetilde{\mathbf{K}}_1 \\ \widetilde{\mathbf{K}}_2 \\ \vdots \\ \widetilde{\mathbf{K}}_n \end{bmatrix}_{(6n\times 6)} \qquad \widetilde{\mathbf{C}}^G = \begin{bmatrix} \widetilde{\mathbf{C}}_1 \\ \widetilde{\mathbf{C}}_2 \\ \vdots \\ \widetilde{\mathbf{C}}_n \end{bmatrix}_{(6n\times 6)}$$

where for the ith point in the build area we define:

$$\widetilde{\mathbf{K}}_i := \mathbf{K}_i\mathbf{\Gamma} \;\;, \widetilde{\mathbf{C}}_i := \mathbf{C}_i\mathbf{\Omega}$$

We take the singular value decompositions of both $\widetilde{\mathbf{K}}^G$ and $\widetilde{\mathbf{C}}^G$ to obtain:

$$\widetilde{\mathbf{K}}^G = \mathbf{U}_f^G\mathbf{\Sigma}_f^G\mathbf{V}_f^{GT}, \;\; \widetilde{\mathbf{C}}^G = \mathbf{U}_\gamma^G\mathbf{\Sigma}_\gamma^G\mathbf{V}_\gamma^{GT}$$

The first three values on the diagonal of the middle term, $\mathbf{\Sigma}^G$, provide three scalars which quantify the overall linear stiffness or angular compliance of all n points in the build area. Let the three scalars on the diagonal of $\mathbf{\Sigma}_f^G$ be, in descending order of magnitude, $\sigma_{1,f}$, $\sigma_{2,f}$, and $\sigma_{3,f}$. Define $\sigma_{1,\gamma}$, $\sigma_{2,\gamma}$, and $\sigma_{3,\gamma}$ simmilarly for $\mathbf{\Sigma}_\gamma^G$. We can then state the four stiffness metrics suggested by Courtielle et al. and used as the basis of our stiffness performance function in this work. They are:

$$S_{k1} = \frac{\sigma_{1,f}}{\sigma_{3,f}}, \qquad S_{k2} = \sigma_{3,f}$$
$$S_{c1} = \frac{\sigma_{1,\gamma}}{\sigma_{3,\gamma}}, \qquad S_{c2} = \sigma_{3,\gamma}$$

In this work, our task is to combine these stiffness metrics into a single performance function that corresponds to the milling task in the reconfigurable design problem. We use a simple weighted sum of all four terms to provide this performance function. In future work, this function might be replaced by a higher-fidelity computer simulation of a milling job, or a utility function validated using simulation. For the purposes of the current work, the simple weighted sum is sufficient to demonstrate the design method. In the sum, metrics S_{k2} and S_{c2} are normalized by nominal global values to bring them on the same order as S_{k1} and S_{c1}. Thus we can write our stiffness performance function as:

$$f^1 = S_{k1} + S_{c1} - \frac{S_{k2}}{S_{k2,nom}} + \frac{S_{c2}}{S_{c2,nom}}$$

Note that the best performance comes from minimizing this performance function. By minimizing the first two terms, S_{k1} and S_{c1}, we attempt to obtain a more uniform stiffness in the build area. The second two terms, are meant to maximize the overall level of linear stiffness and minimize the overall level of angular compliance, respectively.

Speed Performance Function

The speed performance function of a given design is calculated using the kinematic Jacobian at a set of points in the build area. This is a new metric introduced here. The kinematic Jacobian can be written as:

$$J = \begin{bmatrix} \frac{\partial z_1}{\partial x_e} & \frac{\partial z_1}{\partial y_e} & \frac{\partial z_1}{\partial z_e} \\ \frac{\partial z_2}{\partial x_e} & \frac{\partial z_2}{\partial y_e} & \frac{\partial z_2}{\partial z_e} \\ \frac{\partial z_3}{\partial x_e} & \frac{\partial z_3}{\partial y_e} & \frac{\partial z_3}{\partial z_e} \end{bmatrix}, \tag{2.7}$$

where x_e, y_e, and z_e are the Cartesian coordinates of the end effector, and z_1, z_2, and z_3 are the coordinates of the three carriages. The analytical expressions of each entry in the Jacobian expression can be easily obtained by differentiating the following inverse kinematic equation:

$$z_i = \sqrt{L^2 - (x_i - x_e)^2 - (x_i - x_e)^2} + z_e, \tag{2.8}$$

where x_i, y_i and z_i are the coordinates of the carriage, and L is the strut length for the design. This Jacobian relates input velocities at the carriages into end effector velocity by the following relation:

$$J \begin{bmatrix} \frac{\partial x_e}{\partial t} \\ \frac{\partial y_e}{\partial t} \\ \frac{\partial z_e}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial z_1}{\partial t} \\ \frac{\partial z_2}{\partial t} \\ \frac{\partial z_3}{\partial t} \end{bmatrix} \tag{2.9}$$

Provided the Jacobian is invertible, we can multiply both sides by J^{-1} to obtain:

$$\begin{bmatrix} \frac{\partial x_e}{\partial t} \\ \frac{\partial y_e}{\partial t} \\ \frac{\partial z_e}{\partial t} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial z_1}{\partial t} \\ \frac{\partial z_2}{\partial t} \\ \frac{\partial z_3}{\partial t} \end{bmatrix} \tag{2.10}$$

Thus, at each point in the build envelope, the inverse Jacobian maps carriage velocity to end effector velocity. At a given point, if we let carriage velocity vary along all possible directions, the resulting end effector velocity vectors will fall along a 3-dimensionsional ellipse. The minimum principal axis of this ellipse shows the limiting velocity direction and magnitude.

This limiting velocity forms the basis of our speed metric. Over a set of points in the build area, we observe the minimum limiting velocity. For point i in the build envelope let the limiting velocity magnitude be λ_i. The performance function to be minimized is:

$$f^2 = -(\min_i \lambda_i)$$

Cost Performance Function

Incorporation of cost of reconfigurability is a critical part to this case study, as it establishes the expanded Pareto set introduced in Section 2.4. Olenik et al. described a few possible sources of cost in a reconfigurable system [1]. We consider two sources of costs, fixed and variable. Fixed costs are associated with making a variable in the design reconfigurable, while variable costs are associated with the extent of changeability. Thus, the cost of reconfigurability of design variable i can be written as:

$$c_i = \begin{cases} 0 & \Delta x_i = 0 \\ FC + VC(\Delta x) & \Delta x_i \neq 0 \end{cases}$$

where we define Δx_i as:

$$\Delta x_i = \max_k(x_i^k) - \min_k(x_i^k)$$

This nonlinear function is a fair representation of the cost of reconfigurability, but is discontinuous. This type of discontinuity also arises in product family design research in the variable selection problem (e.g., see the product family design problem formulation given in Section 1). In an effort to model this discontinuity in an objective function, we will use a continuous mapping function as an approximation. The details of this mapping implementation are left for Section 2.6.2.

2.6 Case Study Implementation and Results

For this study, a polar grid with 10 equally spaced radial steps (from 0 to R_{max} radians) and 36 equally spaced angular steps (from 0 to 2π) was used to determine global metrics. Symmetry of the build area was exploited to reduce the number of points necessary in the grid by a factor of 6. Using this set of points in the build area, we used a multiobjective genetic algorithm to generate an extended Pareto front. This extended Pareto front was generated for varying values of fixed and variable cost of reconfigurability. Details of the multiobjective genetic algorithm and cost parameter sweep are given next.

2.6.1 Multiobjective Genetic Algorithm Parameters

In order to determine the Pareto set of the reconfigurable system design problem, a multiobjective genetic algorithm was used. This implementation provides reasonable means to determine the Pareto set. The values used within the genetic algorithm are included for completeness, but we emphasize that they are only a means to the objective of this work, and not central to its contribution. The genetic algorithm was carried out in the MATLAB® programming environment using the built-in genetic algorithm function with the options specified in Table 2.2.

Table 2.2: Genetic Algorithm Parameters Used

Parameter	**Value**
Population Size	300
Initial Population Dist.	Uniform
Crossover Fraction	0.4
Pareto Fraction	0.35
Elite Count	30

2.6.2 Modeling Nonlinearity in Cost Objective

Modeling the non-linearity of the cost objective was accomplished using a sigmoid function. This function provides a convenient method to approximate a discontinuous function with a continuous one. The sigmoid function is an s-shaped curve that transitions continuously from 0 to 1. The location of that transition and its steepness is controlled by a shifting parameter, α, and scaling parameter, β, respectively.

$$\sigma(t, \alpha, \beta) = \frac{1}{1 + e^{\beta(x-\alpha)}}$$

Figure 2.9 displays several sigmoid curves for $\alpha = 0$ and varying values of β.

2.6.3 Numerical Experiments

The set of points used to determine the global performance metrics were taken from a polar grid of points sampled for 10 evenly spaced radii at 10 degree increments. (This polar grid of sample points was used for all subsequential numerical experiments as well.) The first attempt to find the Pareto front of this reconfigurable design problem was performed using the values of fixed cost (FC), variable cost (VC), α, and β that are given in Table 2.3. The

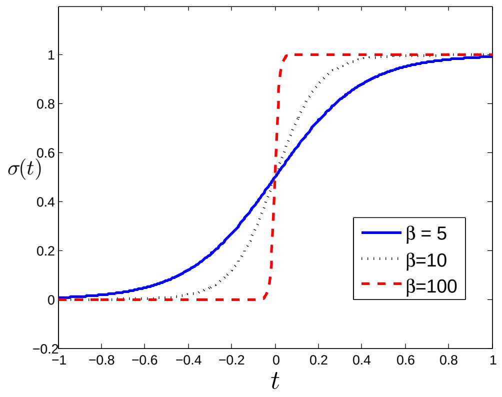


Figure 2.9: Sigmoid curves over varying values of β

fixed costs for each variable was taken to be 1 in this base case, and the variable costs were set such that the cost of varying either variable along the full range given in Table 2.1, would be 1.

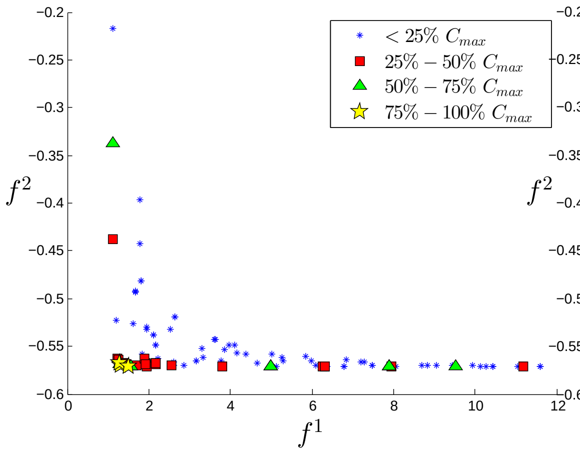
Table 2.3: Baseline costs used in generation of extended Pareto front

D		L			
FC	VC	FC	VC	α	β
1	0.501	1	3.33	0.05	100

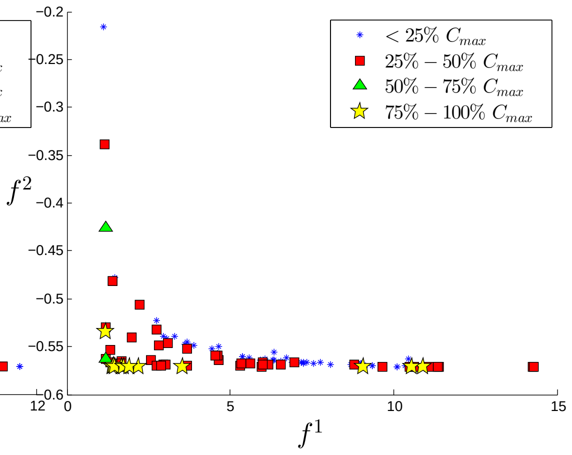
After obtaining a Pareto front from this case study, a numerical experiment was performed to measure the effect of fixed and variable cost values on the determined Pareto front. In this parameter sweep the ratio of fixed to variable cost was varied while the values of sigmoid parameters were held constant to those in the base case. In the 10 runs of the genetic algorithm, variable costs were held constant at the base level while variable costs were increased up to 10 times its base value. In another 10 runs, the experiment was reversed. Fixed cost levels were held constant as variable costs were increased to up to 10 times their level in the base case.

2.6.4 Results

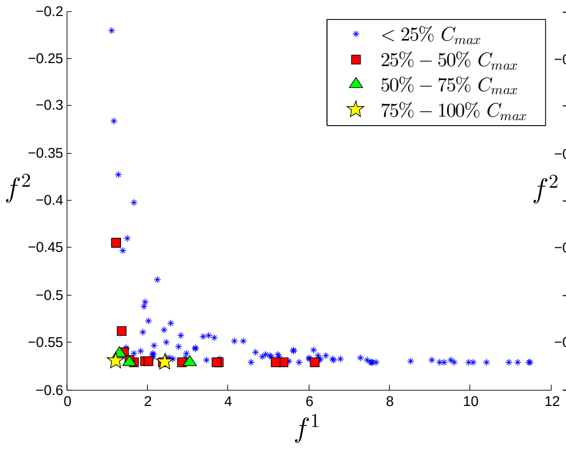
The case study resulted in the determination of over 100 points distributed along the expanded Pareto surface for each genetic algorithm run. Selected



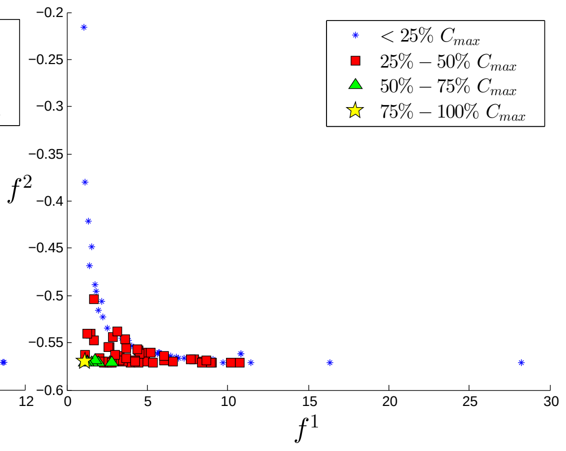
(a) Expanded Pareto surface:
$VC = VC_{base}$, $FC = 1$



(b) Expanded Pareto surface:
$VC = VC_{base}$, $FC = 9$



(c) Expanded Pareto surface:
$VC = 10VC_{base}$, $FC = 1$



(d) Expanded Pareto surface:
$VC = 1VC_{base}$, $FC = 8$

Figure 2.10: Selection of extended Pareto fronts found in case study. Different markers show attainable performance at different levels of cost. C_max is the highest cost of a system in the study.

expanded Pareto surfaces that were obtained are included in figure 2.10.

Notably, these four expanded Pareto sets bear strong resemblance to the surface shown in Fig. 2.5. The lowest cost Pareto set shows the attainable performance of designs with little or no reconfigurability. As solutions become more costly, they approach the performance of the fixed system's utopia point. These figures show how the expanded Pareto surface of the same reconfigurable design problem changes as the relative magnitudes of fixed and variable cost of reconfigurability are altered.

At an early stage in design, having access to points in this performance space helps to demonstrate the tradeoff between performance and cost present in the system. Being able to select from a set of near-Pareto optimal reconfigurable designs circumvents the difficulty of comparing the full Pareto sets associated with each reconfigurable design. This helps to streamline the reconfigurable design problem in the case of offline continuously reconfigurable systems designed for multiability.

2.7 Conclusions and Future Work

We have shown that the variable selection problem in a particular subset of reconfigurable system design can be framed as a variant of the product family design variable selection problem. Further, this formulation allows the Pareto set of the reconfigurable problem to be obtained with the addition of only one dimension onto the performance space.

At early stages of design, having access to the expanded Pareto surface would be a great asset to any designer. For the addition of a single dimension to the performance space, this surface contains significantly more information than the static system's Pareto front. For the continuously reconfigurable systems designed for multiability, a designer can distill the pertinent information from all attainable m-dimensional Pareto sets into a single expanded Pareto surface of dimension $m+1$, where each point in the surface represents a Pareto optimal design along the spectrum of the commonality-performance tradeoff.

While this design formulation is useful for this class of reconfigurable system, the method used to determine the Pareto set in this paper was prepackaged and not specifically tuned for its purpose. One potential area for future development of this work is the selection of a more targeted Pareto discovery method.

The case study used in this work is also an important area of future development. It is a stepping stone toward designing a fully reconfigurable Delta robot capable of functioning as a versatile manufacturing tool. This initial stage of research provides a useful method to obtain non-dominated designs in reconfigurable design problem, but the cost and utility models used will be improved in future work.

Future plans to improve utility models involve the use of simulation to better capture the design requirements of the selected manufacturing operations. These simulations could serve as performance functions directly, or might form the basis of lower fidelity, higher-speed surrogate performance functions. Also, the equations used in this study for the cost of reconfigurability were not based on any industrial cost model. These cost functions were used only to demonstrate the expanded Pareto front concept. Future development of realistic cost models would give the case study results greater weight in the actual fabrication of a reconfigurable Delta robot.

CHAPTER 3

DYNAMIC SYSTEM DESIGN

3.1 Motivation

The case study in Chapter 2 demonstrated the usefulness of the extended Pareto front in the design of a reconfigurable Delta robot for multiability. However, there were approximations used in this case study that could be made more accurate in future work. These approximations included the assumption that the design metrics for speed and rigidity accurately reflected the performance of the Delta robot at the two tasks in question. The speed metric was based strictly on the kinematics of the design, while the rigidity metric was based on a linear finite element model of rigidity of the end effector about a fixed set of points in the build envelope. Neither of these metrics took into account the dynamics of the Delta robot.

This particular gap, i.e., the fact that the Delta robot performance metrics did not take into account system dynamics, motivated a research effort to model the dynamics of compliant members, and attempt to control them. Derivation of a full-scale model for the compliance in the Delta robot was unrealistic for the scope of this thesis, however a smaller effort to model and control the compliance in a single member was undertaken. This effort is a building block toward the longer term goal of creating higher-fidelity performance functions for the different tasks accomplished by the Delta-robot-based multimanufacturing tool.

The following section details the derivation of dynamic equations for a single compliant member that is pinned at one end and free at the other. This derivation follows that given in [39], with some differences.[1] After deriving the equations for a compliant link, we introduce the optimal control design

[1]It is worth noting that upon further examination of [39], it was discovered that the dynamic equation for a compliant link (given in the authors' equation (4)) is actually missing a term. We have corrected the error in this work.

method called direct transcription. In the next chapter we use the dynamic model developed here with direct transcription to perform simultaneous plant and control design.

3.2 Deriving Equations of Motion for a Compliant Link

Energy methods are now applied to derive the equations of motion of an actively-controlled compliant link. One side of the flexible link is attached to a rotary input at which torques can be applied, the other end is free. The link is assumed to be inextensible, and all motions occur in a 2-dimensional plane of rotations under the influence of gravity. With some differences in application, we use the same approach as in [39] to discretize the compliant member and model its dynamics. This method is often referred to as the method of assumed modes. For a derivation of the energy methods used in this chapter, i.e. the Euler-Lagrange equation, see Appendix A.

The deflection of the member at time t can be described by the angle of the tangent vector along the length of the deformed member, $\theta(s,t)$ (where s is taken as the length along the member from the hinge joint). An image of the compliant link is shown in Fig. 3.1. Note that the tangent vector and normal vector are denoted $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, respectively, and can be easily written out in terms of tangent angle θ as shown in equations 3.1 and 3.2.

$$\boldsymbol{\beta(s,t)} \triangleq \begin{bmatrix} cos(\theta(s,t)) \\ sin(\theta(s,t)) \end{bmatrix} \tag{3.1}$$

$$\boldsymbol{\gamma(s,t)} \triangleq \begin{bmatrix} -sin(\theta(s,t)) \\ cos(\theta(s,t)) \end{bmatrix} \tag{3.2}$$

We begin by writing an expression for the kinetic energy and potential energy of the member. Afterward, the method of assumed modes is used to rewrite kinetic and potential energy expressions as a function of a finite number of kinematically independent coordinates (i.e. the generalized coordinates). The expressions for kinetic and potential energy are substituted into the Euler-Lagrange equation to derive dynamic equations that describe

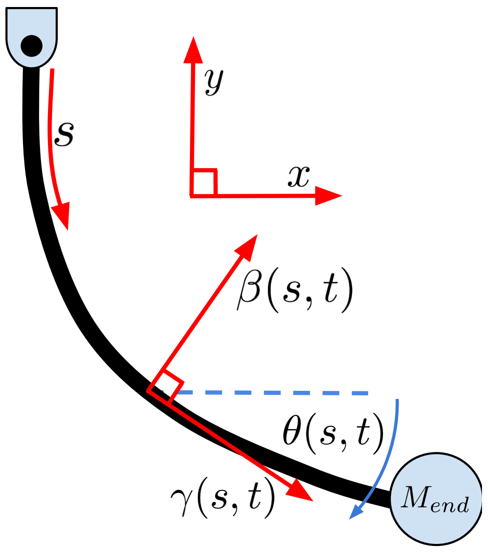


Figure 3.1: A compliant link. The tangent vector β and normal vector γ are measured with respect to the positive x direction, and vary with both time and the convected coordinate s. The link is actuated by a torque input at its hinged base.

the evolution of system coordinates with time.

We can write the kinetic energy in the deformed member at time t as:

$$KE(t) = \frac{1}{2}\int_0^\ell \rho_L \left(\begin{bmatrix}\dot{x}(s,t) & \dot{y}(s,t)\end{bmatrix}\right) \cdot \left(\begin{bmatrix}\dot{x}(s,t) & \dot{y}(s,t)\end{bmatrix}\right) ds \tag{3.3}$$

where ρ_L is the linear mass density along the member.

If we assume that all the strain energy in the member originates from bending, we can write the strain energy in the member at time t as:

$$SE(t) = \frac{E}{2}\int_0^\ell I(s) \left(\frac{\partial \boldsymbol{\beta}(s,t)}{\partial s(s,t)}\right) \cdot \left(\frac{\partial \boldsymbol{\beta}(s,t)}{\partial s(s,t)}\right) ds \tag{3.4}$$

where E is the elastic modulus of the beam material (assumed constant), and I is the moment of inertia of the cross section with respect to the bending axis.

Finally, we can also write the gravitational potential energy of mass along the length of the beam as:

$$GPE(t) = \int_0^{\ell} \rho_L \; g \int_0^{s} (\boldsymbol{\beta}(\hat{s},t) \cdot \hat{\mathbf{j}}) d\hat{s} \; ds + gM_{end} \int_0^{\ell} \boldsymbol{\beta}(s,t) \cdot \hat{\mathbf{j}} ds \tag{3.5}$$

where $\hat{\mathbf{j}}$ is a unit vector in the y direction of the inertial frame.

The next step is spatial discretization. We divide the beams of our system into elements, and write the continuous unit tangent vector $\boldsymbol{\beta}(s,t)$ in terms of first order shape functions (p_n), and the value of $\boldsymbol{\beta}$ at node n:

$$\boldsymbol{\beta}(s,t) = \sum_{n=1}^{\#\text{nodes}} \boldsymbol{\beta}_n(t) p_n(s) \tag{3.6}$$

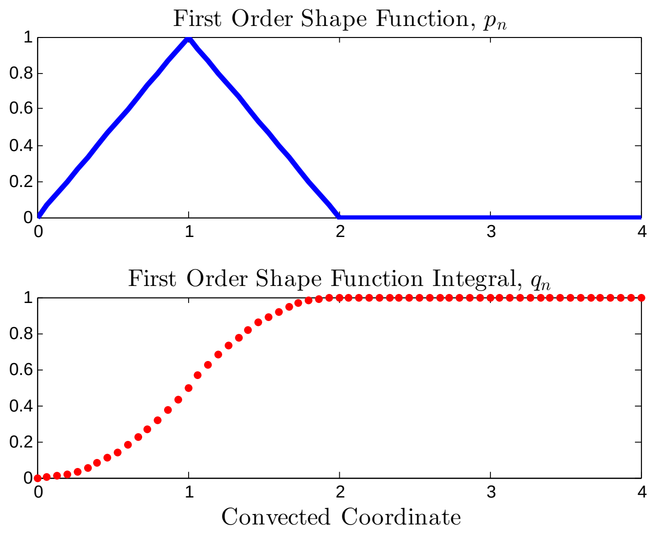


Figure 3.2: An example first order shape function $p_n(s)$ and its integral $q_n(s)$ for a node centered at coordinate 1. Note that nodes have unit spacing.

This discretization allows us to rewrite the system kinetic energy, strain energy, and potential energy as summations of discrete terms. By substituting Eq. 3.6 into Eqs. 3.3, 3.4, and 3.5, we arrive at expressions for kinetic energy, strain energy, and potential energy that depend only on the tangent angles at each node along the length of the beam. These are written in equations 3.7b through 3.9b.

$$KE(t) = \tag{3.7a}$$

$$\tfrac{1}{2}\textstyle\sum_{n=1}^{\#\text{nodes}}\sum_{m=1}^{\#\text{nodes}} \dot{\theta}_m(t)\dot{\theta}_n(t)\gamma_m(t)\cdot\gamma_n(t)\left(\int_0^{\ell}\rho_L\; q_m(s)q_n(s)ds + M_{end}\; q_n(\ell)q_m(\ell)\right) \tag{3.7b}$$

$$= \frac{1}{2}\sum_{n=1}^{\#\text{nodes}}\sum_{m=1}^{\#\text{nodes}} \dot{\theta}_m(t)\dot{\theta}_n(t)\gamma_m(t)\cdot\gamma_n(t)M_{m,n} \tag{3.7c}$$

$$SE(t) = \sum_{n=1}^{\#\text{nodes}}\sum_{m=1}^{\#\text{nodes}} \beta_n(t)\cdot\beta_m(t)\int_0^l \frac{EI(s)}{2}\; p_n'(s)\; p_m'(s)ds \tag{3.8a}$$

$$= \sum_{n=1}^{\#\text{nodes}}\sum_{m=1}^{\#\text{nodes}} \beta_n(t)\cdot\beta_m(t)K_{m,n} \tag{3.8b}$$

$$GPE(t) = g\sum_{n=1}^{\#\text{nodes}} sin(\theta_n)\int_0^l \rho_L(s)p_n(s)ds + M_{end}\sum_{n=1}^{\#\text{nodes}} sin(\theta_n)q_n(L) \tag{3.9a}$$

$$= g\sum_{n=1}^{\#\text{nodes}} sin(\theta_n)\left(\int_0^l q_n(s)\rho(s)ds + M_{end}\; q_n(l)\right) \tag{3.9b}$$

Note that we have made the following substitutions:

$$M_{m,n} \triangleq \int_0^{\ell} \rho_L q_m(s)q_n(s)ds + M_{end}\; q_n(\ell)q_m(\ell) \tag{3.10a}$$

$$K_{m,n} \triangleq \int_0^l \frac{EI(s)}{2}\; p_n'(s)\; p_m'(s)ds \tag{3.10b}$$

$$q_m(s) \triangleq \int_0^s p_m(\hat{s})d\hat{s} \tag{3.10c}$$

The quantities $M_{m,n}$ and $K_{m,n}$ above can be intuitively interpreted as entries in a system mass matrix $\mathbf{M}$ and system stiffness matrix $\mathbf{K}$. It is very important to note that the system mass matrix, stiffness matrix, and shape functions do not vary in time. Thus, these quantities need to be calculated only once before the differential equation can be solved. This is a great advantage of using the method of assumed modes.

Having expressions for the kinetic energy, strain energy, and gravitational potential energy as functions of a finite set of state variables, we can use the Euler-Lagrange equation to determine a set of dynamic equations governing the evolution of each state, i.e., all θ_ns, along the length of the link.

The Euler-Lagrange equation for generalized coordinate q_j is given by:

$$\frac{d}{dt}\left(\frac{\partial(T-V)}{\partial \dot{q}_j}\right) - \frac{\partial(T-V)}{\partial q_j} - Q_{j,nc} = 0 \tag{3.11}$$

Substituting the right hand side of equation 3.7c for T, the sum of the right hand sides of equations 3.8b and 3.9b for V, and replacing generalized coordinate q_j with θ_j we obtain a differential equation for each θ_j. Let $L \triangleq T - V$, we can write it as:[2]

$$L = \frac{1}{2}\sum_{m=0}^{\#\text{nodes}}\sum_{n=0}^{\#\text{nodes}} \dot{\theta}_m\dot{\theta}_n\gamma_m\cdot\gamma_n M_{m,n}$$
$$-\frac{1}{2}\sum_{m=0}^{\#\text{nodes}}\sum_{n=0}^{\#\text{nodes}} \beta_m\beta_n K_{m,n} - g\sum_{n=1}^{\#\text{nodes}} sin(\theta_n)\left(\int_0^\ell q_n(s)\rho(s)ds + M_{end}\ q_n(\ell)\right) \tag{3.12}$$

Taking the derivative of L with respect to generalized coordinate θ_n we obtain:

$$\frac{\partial L}{\partial \theta_n} = -\sum_{m=0}^{\#\text{nodes}} \dot{\theta}_m\dot{\theta}_n\gamma_m\cdot\beta_n M_{m,n} - \sum_{m=0}^{\#\text{nodes}} \beta_m\cdot\gamma_n K_{m,n}$$
$$- g\ cos(\theta_n)\left(\int_0^\ell q_n(s)\rho(s)ds + M_{end}\ q_n(\ell)\right) \tag{3.13}$$

Taking the derivative of L with respect to $\dot{\theta}_n$ we obtain:

[2]Those familiar with calculus of variations, optimal control, or Hamilton's principle may recognize the use of the symbol L to refer to the difference $T - V$ in a mechanical system. In this context, the variable L is used to refer to the *Lagrangian*. In more general contexts, the Euler-Lagrange equation (less the $Q_{j,nc}$ term) can be shown to be a necessary condition to minimize the cost of some trajectory $y(x)$ from 0 to T, where the cost J of the trajectory is given as $\int_0^T L(x, y, y')dx$. A good reference for this material is [40].

$$\frac{\partial L}{\partial \dot{\theta}_n} = \sum_{m=0}^{\#\text{nodes}} \dot{\theta}_m \gamma_m \cdot \gamma_n M_{m,n} \tag{3.14}$$

Subsequently taking the time derivative of equation 3.14 yields:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_n}\right) = \sum_{m=0}^{\#\text{nodes}} \ddot{\theta}_m \gamma_m \cdot \gamma_n M_{m,n} + \sum_{m=0}^{\#\text{nodes}} \dot{\theta}_m M_{m,n} \left(-\beta_m \dot{\theta}_m \cdot \gamma_n - \beta_n \dot{\theta}_n \cdot \gamma_m\right) \tag{3.15}$$

The Euler-Lagrange equation for the nth generalized coordinate in this system is thus:

$$\sum_{m=0}^{\#\text{nodes}} \left[\ddot{\theta}_m \gamma_m \cdot \gamma_n M_{m,n} + \dot{\theta}_m M_{m,n} \left(-\beta_m \dot{\theta}_m \cdot \gamma_n - \beta_n \dot{\theta}_n \cdot \gamma_m\right) + \beta_m \cdot \gamma_n K_{m,n}\right] + g\ cos(\theta_n) \left(\int_0^l q_n(s)\rho(s) ds + M_{end}\ q_n(l)\right) = Q_{n,nc} \tag{3.16}$$

Note that in this problem, the generalized non-conservative force $Q_{n,nc}$ is simply the torque applied to node n.

Equation 3.16 gives one equation for each state of the system. Each equation is not yet in explicit form, because $\ddot{\theta}_n$ is not equated to some function of lower order derivatives. However, because all $\ddot{\theta}_m$ terms appear linearly, the n system equations can be solved at every timestep explicitly for each $\ddot{\theta}_n$ using simple linear algebra.

3.3 Co-design of a Compliant Link with Direct Transcription

Having a dynamic model of a compliant link, the next step toward simulating a system with compliant members was to perform control design. In the larger picture, lessons learned in the simpler single-member problem might then be translated into the more complicated situation of the Delta-robot-based multimanufacturing tool in order to develop higher fidelity performance metrics.

Commercial programs do exist that can perform dynamic simulation of the Delta robot given a particular control law or control trajectory. And while incorporating co-design is possible by iteratively simulating these dynamic models with different plant and control variables, solving the problem using this nested optimization is more vulnerable to numerical instabilities [41,42]. Besides being less sensitive numerically, the optimization problem, co-design through direct transcription can take advantage of parallel computing.

In the remainder of this section, we discuss one method used to perform co-design, direct transcription (DT). Essentially, the plant and control design problem is transcribed into a large nonlinear program (NLP) with a sparse structure. Sparse nonlinear optimization algorithms (in our case SNOPT) are then used to try and find a solution to the nonlinear program. By solving the nonlinear program, a designer can solve the original optimal control problem.

Direct transcription can be used in concert with co-design, because once the continuous optimization problem has been transcribed to discrete form, additional design variables can be added to the problem. Solving the NLP then amounts to simultaneously solving the control and structural design (i.e. the co-design) problem. In the case of the Delta, solving the co-design problem would allow us to prescribe a control trajectory for a given task, along with an optimal strut length. In addition, direct transcription supports the inclusion of inequality design constraints, which are critical for the realistic treatment of physical design considerations in a co-design problem [42].

The following subsection is meant to familiarize the reader with the concept of direct transcription. It demonstrates the concept before we talk about the results of utilizing the method in relation to the co-design of a compliant member.

3.3.1 Direct Transcription Formulation

Direct transcription (DT) is an all-at-once (AAO) discretize-then-optimize method of optimal control that transforms a continuous control design problem into a finite-dimensional nonlinear optimization problem (NLP). We now provide a brief description of the method to familiarize the reader with its use.[3]

[3]The interested reader is referred to [41] and [43] for more general background on direct transcription.

Consider the following generalized optimal control problem:

$$\begin{aligned} \min_{\boldsymbol{\xi}(t),\mathbf{u}(t),\mathbf{x}_p} \quad & \int_0^{t_F} L(\boldsymbol{\xi}(t),\mathbf{u}(t),\mathbf{x}_p)dt \\ \text{where} \quad & \dot{\boldsymbol{\xi}} = \mathbf{f_d}(\boldsymbol{\xi}(t),\mathbf{u}(t),\mathbf{x}_p,t) \\ & \boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 \end{aligned} \tag{3.17}$$

In this formulation, $\boldsymbol{\xi}(t)$ is the system state trajectory, $\mathbf{u}(t)$ is the control trajectory, $\mathbf{x}_p$ is the vector of physical system design variables, and $\mathbf{f_d}$ is the state-space time derivative function that models system dynamics. The function $L(\cdot)$ refers to the running cost (or cost incurred per unit time) associated with a given control and state trajectory.[4] Direct transcription transcribes all continuous aspects of the original formulation into algebraic constraints. The system state and control trajectories are discretized in time, and the state equations are converted from continuous differential equations to a system of algebraic constraints using a collocation method [41].

The optimization formulation given in formulation 3.17 can be transcribed into the following finite-dimensional NLP, which may be easier for a designer to solve:

$$\begin{gathered} \min_{\mathbf{U},\boldsymbol{\Xi},\mathbf{x}_p} \quad \sum_{i=1}^{n_t-1} L(\boldsymbol{\xi}_i,\mathbf{u}_i,\mathbf{x}_p)h_i \\ \text{subject to:} \\ \boldsymbol{\zeta}_i(\mathbf{U},\boldsymbol{\Xi},\mathbf{x}_p) = -\frac{h_i}{2}\left(\mathbf{f_d}_i + \mathbf{f_d}_{i+1}\right) + \boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i = \mathbf{0}, \qquad i = 1,2,\ldots,n_t-1 \end{gathered} \tag{3.18}$$

The variables $\mathbf{U}$ and $\boldsymbol{\Xi}$ are matrices that are discretized representations of the control and state trajectories. The the ith row of these matrices correspond to the control and state trajectory values at the ith timestep, respectively. The function $\boldsymbol{\zeta}_i(\cdot)$ is found by applying a collocation method to the state equations of the original continuous formulation. $\boldsymbol{\zeta}_i(\cdot)$ approximately enforces the dynamic equations of the continuous system through algebraic constraints. The summation term approximates the cost of a control and state trajectory by using the value of the Lagrangian function only at the finite number of timesteps. The variable h_i in this summation is the time

[4]The reader may recognize the use of the Lagrangian from the derivation of energy methods from Section 3.2. The Lagrangian used here is related but distinct from the Lagrangian used in energy methods.

duration of the ith time step. In this example, the trapezoidal method is used as a simple implicit collocation method, but many other higher order methods are possible.

Because the control and structural optimization problem has been converted to a nonlinear program, additional constraints that are present in the original continuous problem can be added to the transcribed problem. Constraints on states being applied to elements of $\mathbf{\Xi}$, and constraints on control being applied to elements of $\mathbf{U}$ [43]. Solving the co-design problem entails solving this single NLP. It is a large-scale problem due to the large number of optimization variables and constraints, but its sparse problem structure can be exploited for efficient solution, and it supports fine-grained parallel computing.

Focus now shifts to application of the approaches developed in the previous chapter. We begin applying direct transcription and a model of the simple compliant manipulator to a simultaneous control and structural design problem.

CHAPTER 4

APPLIED DYNAMIC SYSTEM DESIGN: CO-DESIGN OF A SIMPLE COMPLIANT MANIPULATOR

4.1 Introduction

Having discussed the mathematical model of the compliant link and the use of direct transcription to perform co-design, focus now is directed toward application—the co-design of simple robotic manipulator comprised of a single compliant link. A depiction of the single compliant link is given in figure 3.1.

Dynamic analysis of the mechanical members of robotic manipulators almost always involves one of the following two assumptions, either (1) that link deformation is small enough to permit linear approximation of the system, [44] or (2) that links remain perfectly rigid. [45–47] Under these approximations, naturally occurring vibrations in the system are deemed "parasitic" motions that degrade the accuracy of kinematic equations used to model the robot. In practice, the effect of vibrations resulting from the elastic passive dynamics of the system are either avoided by using stiff enough members, and/or suppressed via damping [44].

An interesting prospect in the design of robotic manipulators is utilizing, rather than avoiding, the system natural dynamics during dynamic design. If compliance in robotic members can be accounted for, can certain tasks actually benefit from exploiting natural dynamics? Typical control design regards the system design as immutable, and attempts to optimize performance only by altering control. Where possible, simultaneous control and system design offers the potential to design lighter, faster, and more energy-efficient robotic systems.

In this research effort we use direct transcription to simultaneously design the control system and passive dynamics of a simple robotic manipulator for a predefined pick-and-place task. In previous studies, direct transcription has been shown to be a promising method of simultaneous control and plant design [42,48,49]. As the current work evolves, we begin to explore its efficacy in co-design problems involving the use of compliant members. Other than an interesting application of co-design, this work is also an initial step toward using higher fidelity performance metrics that take into account dynamics of the Delta robot.

In the current application, direct transcription is performed using the GPOPS-II[1] package for solving optimal control problems. This software implements direct transcription through pseudospectral methods (PSMs). PSMS are a specific implementation of DT where single higher-order polynomials are used to represent state trajectories, as opposed to many lower-order polynomials that approximate state trajectories over small time steps. The GPOPS software also automates the generation of collocation points in a MATLAB programming environment. The points are generated and refined by the program using a Legendre-Gauss-Radau quadrature orthogonal collocation method [50].

4.2 System Description

The state vector used in this control design has twice the number of elements as nodes in the system. We can write it as:

$$\boldsymbol{\xi} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \\ \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_N \end{bmatrix} \tag{4.1}$$

Note that θ_i is the tangent angle of the compliant beam at the ith node in the system, and we use N to indicate the total number of nodes.

As has been mentioned during the derivation of the model, the only con-

[1]GPOPS stands for Gauss Pseudospectiral OPtimization Software. It is developed and maintained by Michael A. Patterson and Anil V. Rao of the University of Florida, Gainesville.

trol input used in this study is a torque at the pinned end of the simple manipulator. For the purposes of this co-design study, the derived model of the compliant link has been modified to incorporate viscous damping[2] and to discount gravity. The values of the design parameters used in this study are given in the following table:

Table 4.1: Parameters Used in the Co-design of a Single Compliant Link

Variable	**Symbol**	**Value**
Member in-plane thickness (m)	b	0.012
Member out-of-plane thickness (m)	h	0.02
Payload mass (Kg)	M_{end}	0.1
Control saturation (N·m)	u_{max}	10
Raleigh damping coeff (N·s/m)	ν	0.03
Material elastic modulus (Pa)	E	2×10^9
Material density (Kg/m^3)	ρ	1000
Target point	$\mathbf{p_{target}}$	$\begin{bmatrix} cos(\pi/4) \\ -sin(\pi/4) \end{bmatrix}$

Note that the total length of the member is not mentioned in this table. It is the sole element of the plant design vector $\mathbf{x}_p$. The length of the member is thus changed by the optimization algorithm during the control design.

4.3 Problem Formulation

The first formulations tested through for the co-design approach were the following:

OPTIMIZATION FORMULATION 1: Minimum Time of Travel

[2]Viscous damping is incorporated using a Rayleigh's dissipation function in the energy formulation. This method is detailed on pages 23 and 24 of [51]. If the damping coefficient is ν_1, then the energy dissipated from a node with angular rotation rate $\dot{\theta}$ in time dt is $\nu_1(\dot{\theta})^2 dt$.

in Cyclic Motion

$$\begin{aligned}
\min_{\ell, u(t), \boldsymbol{\xi}(0)} \quad & t_f \\
\text{s.t.} \quad & \dot{\boldsymbol{\xi}}(t) - \mathbf{f}_d(\boldsymbol{\xi}, \mathbf{u}(t), \mathbf{x}_p, t) = \mathbf{0} \\
& \theta_i(0) + \theta_i(tf) + \pi = 0 \quad \forall i \in 1, ..., N \\
& \dot{\theta}_i(0) + \dot{\theta}_i(tf) = 0 \quad \forall i \in 1, ..., N \\
& \|\mathbf{p}_{end}(0) - \mathbf{p}_{target}\|_2 < 0.05 \\
& |u(t)| \leq u_{\max}
\end{aligned} \tag{4.2}$$

OPTIMIZATION FORMULATION 2: Minimum Energy Use in Cyclic Motion[3]

$$\begin{aligned}
\min_{\ell, u(t), \boldsymbol{\xi}(0)} \quad & \int_0^{t_f} u^T u \; dt \\
\text{s.t.} \quad & \dot{\boldsymbol{\xi}}(t) - \mathbf{f}_d(\boldsymbol{\xi}, \mathbf{u}(t), \mathbf{x}_p, t) = \mathbf{0} \\
& \theta_i(0) + \theta_i(tf) + \pi = 0 \quad \forall i \in 1, ..., N \\
& \dot{\theta}_i(0) + \dot{\theta}_i(tf) = 0 \quad \forall i \in 1, ..., N \\
& \|\mathbf{p}_{end}(0) - \mathbf{p}_{target}\|_2 < 0.05 \\
& |u(t)| \leq u_{\max} \\
& t_f \leq 3
\end{aligned} \tag{4.3}$$

[3]The reader may take issue with the second formulation being called "minimum energy." In reality the energy expended by a system could be calculated with the integral $\int_0^{t_f} u^T \dot{\theta}_1(t)$. This formulation suffers from the same numerical difficulties as the minimum time formulation. Optimal control using this objective introduces additional numerical difficulties. It is thus reasonably common to use a better behaved surrogate for energy consumption as in [52, 53].

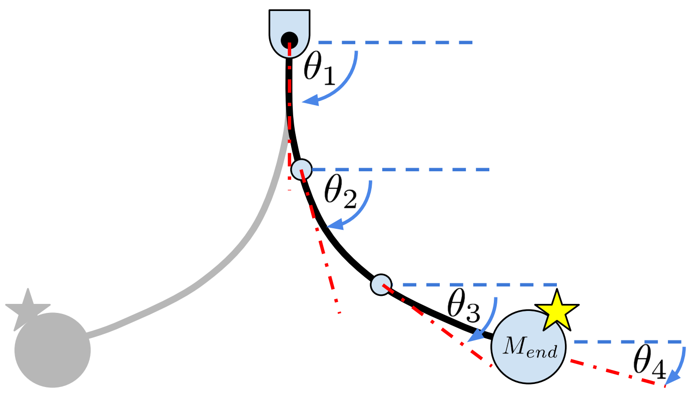


Figure 4.1: The combination of cyclic constraints and a target point constraint forces the end of the compliant member to pass closely (within 5 cm) to the target point and its mirror image in a cyclic motion. This figure displays the compliant member being approximated by four nodes at the beginning of the cyclic motion, along with its mirror image at the end of the motion.

The first constraint given in both formulations ($\dot{\boldsymbol{\xi}}(t)-\mathbf{f}_d(\boldsymbol{\xi},\mathbf{u}(t),\mathbf{x}_p,t)=\mathbf{0}$) ensures that system dynamics are obeyed by the state and control trajectory. The next two constraints, which involve all nodes in the member, are cyclic constraints. They enforce the requirement that whatever motion the compliant member undergoes, it is cyclic. The fourth constraint given in each formulation requires that the position of the end effector $\mathbf{p}_{end}$ is within 5 cm of a target position at the start (and finish) of the cyclic trajectory. In both models we constrain the allowable control effort to be less than u_{max}. Finally, the constraint given in the minimum energy model requires that the total time of the trajectory is less than or equal to 3 seconds.

The first formulation given that minimizes the total time of the trajectory is singular. This can be shown based on the fact that control effort appears linearly in the dynamic equations of the system [41]. While singular optimal control problems can be solved, they can be a source of numerical difficulties. Problems that are singular in this way often display optimal control trajectories that exhibit switching. More information on singular control can be found in [41, 43, 49]. Reference [41] contains an excellent example of singular optimal control with switching on page 213 (the Goddard Rocket Problem).

In the next section, we discuss preliminary results for each optimization formulation. The negative consequences associated with singular control of the single compliant member are also discussed.

4.4 Preliminary Results

Both optimization formulations were implemented in the GPOPS-II programming environment. The following results were obtained using a 3-node model of the compliant member.

4.4.1 Formulation 1

Using direct transcription to solve the co-design problem with minimum time as an objective lead to significant modeling difficulty. The fact that the formulation was singular led to highly oscillatory "bang-bang" type control; this, coupled with the cyclic constraints imposed on the formulation caused feasibility of link trajectories to be difficult to maintain. Once high frequency oscillations were imparted to the system, satisfying cyclic boundary constraints was difficult. The adaptive meshing algorithm used in the GPOPS program terminated prematurely with an infeasible result as a consequence of these numerical difficulties.

One method for simplifying singular optimal control problems is to add a penalty term to the running cost which depends quadratically on the control effort. This approach is termed "quadratic" regularization. Scaling the penalty term appropriately can help find optimal trajectories which are close to the desired optimal control without being part of a singular optimal control problem [41, 49].

4.4.2 Formulation 2

The minimum energy formulation was successfully solved for the nominal parameters listed in Table 4.1. The state and control trajectories are shown in figures 4.2 and 4.3. From these two plots, the state directory can be seen to be almost the same as a perfectly rigid member. The θ and $\dot{\theta}$ values for each node are the nearly identical over all nodes. This behavior satisfies the problem formulation, but does not demonstrate a synergistic relationship between system passive dynamics and control. Any advantages that the more compliant manipulator possessed in terms of energy consumption are only due to its smaller mass, and not its dynamics.

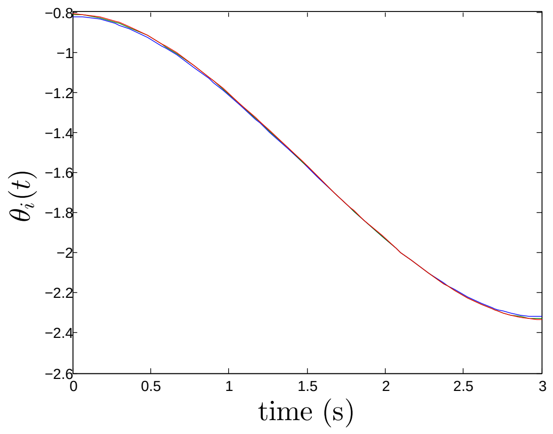


Figure 4.2: The trajectory of system θ values. Note that the three system θ values are almost perfectly superimposed. This behavior is similar to that of a rigid beam.

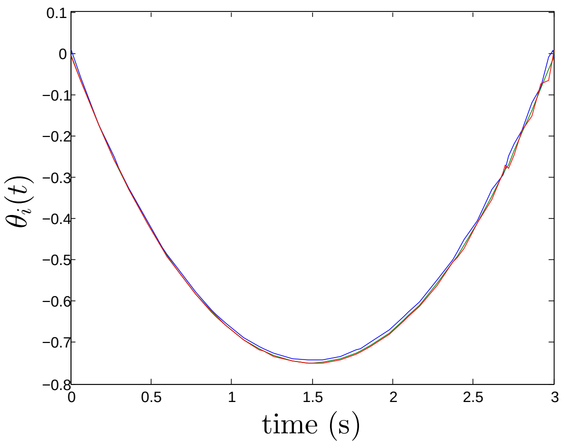


Figure 4.3: The trajectory of system $\dot{\theta}$ values. Note that the three $\dot{\theta}$ values are almost perfectly superimposed in the plot. This behavior is similar to that of a rigid beam.

After obtaining these results, a parametric study was carried out to see if by reducing the compliant member's thickness, solutions to the co-design problem might be found that exhibit deflection without excessive undesired oscillatory behavior. Such an example would show that, for the right plant design, active and passive dynamics could have a more interesting relationship in this problem. Initial efforts to reduce the in-plane thickness of the beam significantly increased the time required to arrive at a terminating condition in the GPOPS algorithm. Even when these analyses were given more time, all exited prematurely returning error messages about numerical difficulties from the nonlinear optimization solver (SNOPT). Even reducing the in-plane thickness h by only one millimeter led to early termination of the GPOPS solver as a result of numerical difficulties. Understanding the sensitivity of the GPOPS solver to differences in compliant system dynamics would be a great subject for future work.

4.5 Conclusions and Future Work

This developing work in co-design of compliant mechanisms provides interesting challenges. How can designers use the benefits of compliance in a mechanical system, i.e., less weight and advantageous natural dynamics, while mitigating the drawbacks that stem from more involved analysis? One difficulty of designing the control of a simple compliant member with direct methods stems clearly from its oscillations and the resulting numerical issues caused by the solver. In a relatively simple 3-node model, we saw that a small increase in member flexibility could easily lead to a large increase in computation time, and nonconvergence of the GPOPS optimal control algorithm. The sensitivity of the problem to design parameters could be mitigated by altering the problem formulation, or shifting the solution approach.

An interesting future case study could look at the effect of increasing the damping coefficient on the convergence time of the co-design result. Increasing the damping coefficient used in the problem would certainly lessen oscillations in the system. This increase in damping, however, will also cause more energy to be wasted by the compliant system, degrading its advantage in terms of energy efficiency.

Attempting to solve the minimum energy formulation presented here using a single-shooting method would provide an informative comparison between co-design methods used in the design of compliant mechanisms. While a single-shooting method would require a complete dynamic simulation between iterations of the optimizer, it would implicitly enforce feasibility during all iterations of the optimization algorithm. Incorporating the cyclic constraint into a single-shooting method would be an interesting challenge.

This study indicates a very fine line between a compliant robotic manipulator capable of being controlled without any noticeable deflections and a compliant system which was so oscillatory that analysis methods struggled to converge to a valid solution. Future work should look at trying to find the intermediate point between these two extremes, or perhaps change the system being modeled so as to widen the line.

CHAPTER 5

DELTA ROBOT DESIGN AND FABRICATION

The actual fabrication of a working reconfigurable Delta robot was an important goal for my Master's degree. The project provided an excellent opportunity to supplement my academic research with hands-on design experience. The Industrial and Systems Engineering Department offered unrestricted access to a Stratasys Dimension Elite 1200$^{\text{TM}}$ 3D printer. I constructed the Delta robot from a mix of 3D printed and off-the-shelf components, using community documentation of best practices as a guide during the design and construction.

The following sections outline the steps taken to go from design to working prototype. Before completion of my Master's thesis, robot operation has not exceeded plotting, but future plans are to build off of the preliminary work done here to build a fully functional multimanufacturing tool.

In the following sections, we (1) define what it means to be a Delta robot, (2) discuss inspirations for the current implementation, (3) outline robot mechanical design/construction, (4) detail software used in the robot's operation, and (5) layout future plans for the machine.

5.1 The Delta Robot

We have already mentioned some of the history of the Delta robot in Section 2.5.1. Here we focus on the functional definition of a Delta robot, and its defining kinematic characteristics. The coming discussion is made clearer by Fig. 5.1, which labels some of aspects of a Delta robot to which we will be referring in further discussion.

A Delta robot necessarily has 3 kinematic inputs, which map to 3 kinematic outputs.[1] While the type of kinematic inputs can vary within the Delta

[1]In figure 5.1 the three kinematic inputs are the linear (vertical) motions of the car-

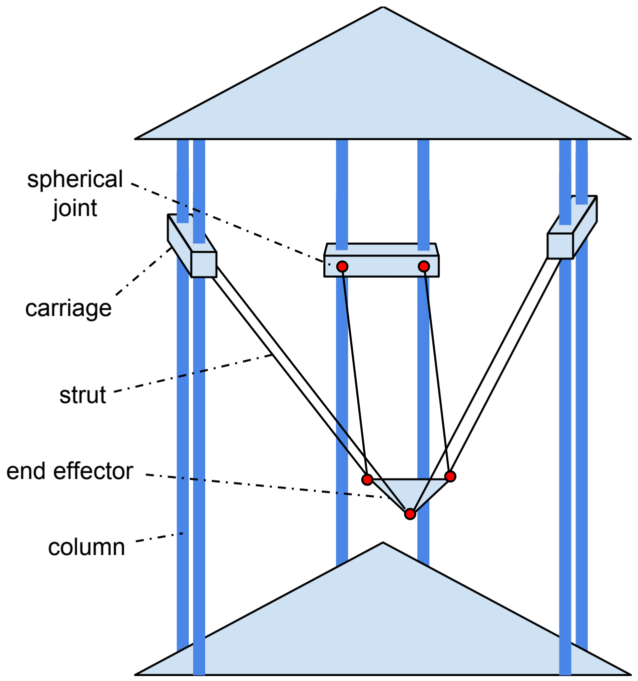


Figure 5.1: A general diagram labeling the parts of a Delta robot

architecture, the kinematic outputs of the Delta are always pure translations by design. This is a defining characteristic of the Delta architecture, and results from a simple geometric condition: two adjacent struts are always parallel.

As discussed in Chapter 4, the advantage of a Delta robot is speed. The actuated mass does not need to include the weight of the actuator, allowing it to be an efficient tool in high-speed pick-and-place operations like placing components on a circuit board or sorting small objects.

5.2 Inspiration

The construction of the Delta robot was largely inspired by the Rostock, a 3D printer designed by Johann in Seattle, Washington in 2012 [54]. This design is shown in figure 5.2. The Rostock seemed to start a trend in 3D printers based on the Delta robot which included multiple Kickstarter Projects (e.g.

riages. In contrast, the kinematic inputs on the ABB Flexpicker (shown in figure 2.6) are angular. Note that in both cases, end effector motion is pure translation.

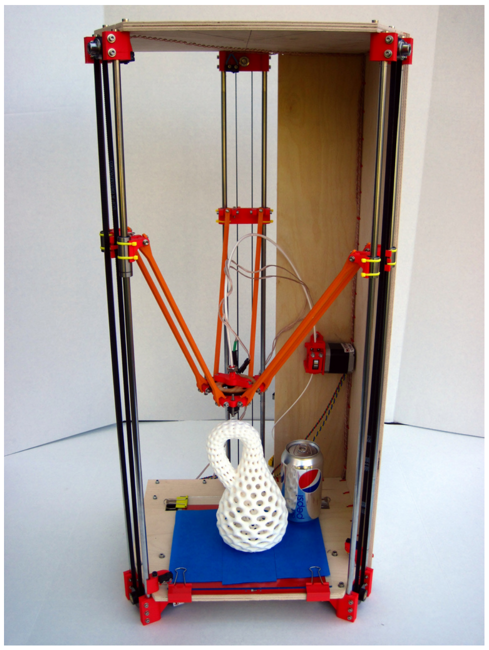


Figure 5.2: The Rostock, a 3D printer based on a Delta robot architecture from [54]

DeltaMaker, Boot Industries' 3D printer, and the Deltaprintr) that each successfully persuaded donors to finance their product.

The decision to build a reconfigurable Delta-robot-based multimanufacturing tool was inspired by this trend, and a desire to connect research work with practical application. The long-term goal was (and still is) to create a Delta robot which could function as not only a 3D printer, but also a light milling machine, a plotter, a pick-and-place robot. The research question beneath these practical applications was how can one systematically design such a product? This question was the driving factor behind the research problem presented in Chapter 2 of this work.

5.3 Construction

The type of input actuation used in our Delta robot was the same as that used in the Rostock—vertical motion of three carriages along linear guides. Additional requirements on our design were (1) struts needed to be resizeable and (2) actuation should be accomplished through DC motors. The first requirement was to make the Delta reconfigurable, the second was was to

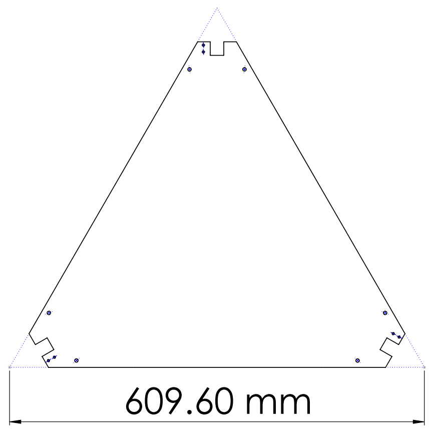


Figure 5.3: Dimensions of steel plate that forms the base and top of our Delta robot. Note that the dimension is measured from the corners of the containing equilateral triangle.

allow for smooth continuous control.

Before outlining the parts used in the Delta robot's construction, it is important to note the overall scale of the robot. The full length of exposed stainless steel columns is 92 cm, and one side of the equilateral triangle at the base of the Delta robot is 609.6 mm (see figure 5.3). The reconfigurable struts are capable of altering the reachable build area, measured in terms of radius from the center of the build envelope, from 50 cm to 175 cm in increments of 25 cm. Note that the maximum circular build area that fits on the robot's build platform has a radius of just over 175 mm ($\sim$ 175.98 mm)

5.3.1 Parts

The final design was constructed out of a combination of off-the-shelf components and 3D printed parts. The 3D printed parts included the carriages, tensioner, end effector, base corners, top corners, joints, and opto-interrupter mounts (see the labels in figure 5.4). Discounting the nuts and bolts used in the design (which were one of three standard metric sizes), all other parts used in the mechanical assembly are listed in Table 5.1.

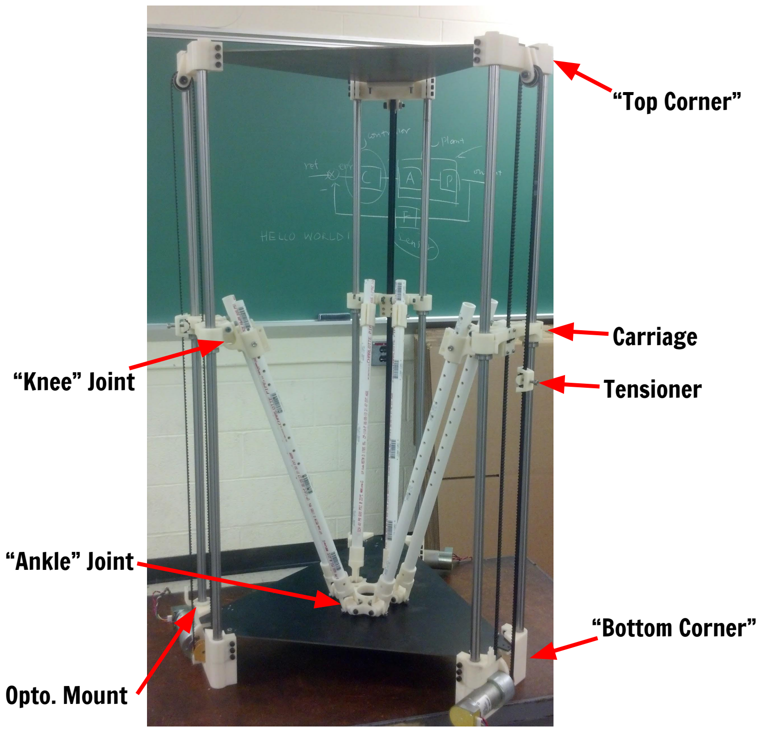


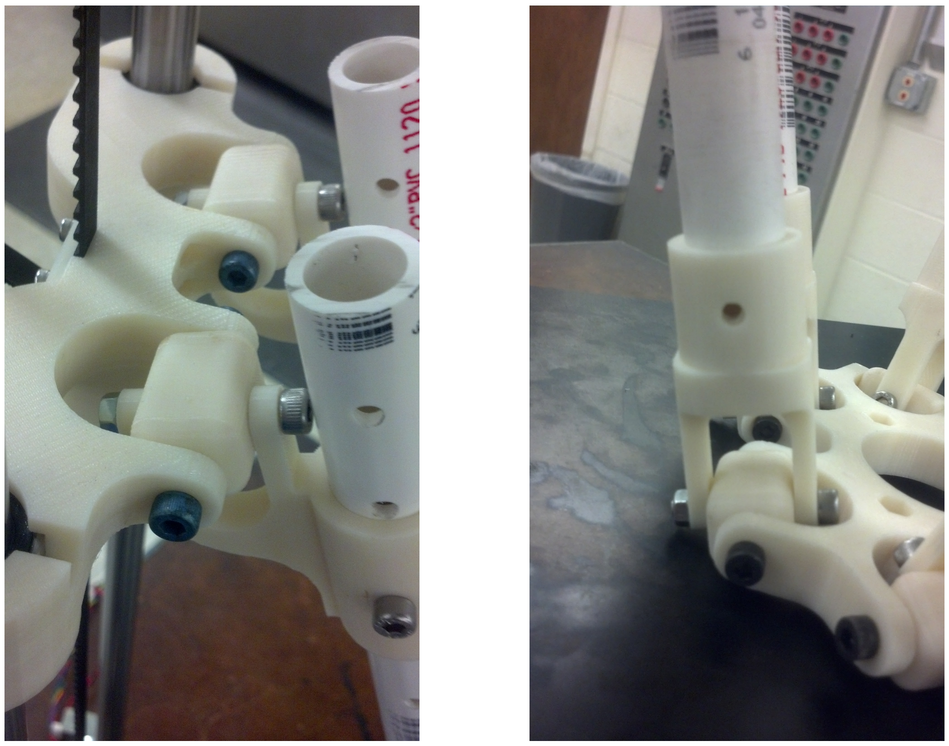
Figure 5.4: Labeled Photo of the fabricated Delta robot

Table 5.1: Nonprinted parts in fabricated delta robot

Part	**Specifications**	**Quantity**
Brushed DC Gearmotor	50 oz-in, 0.25"⌀, 5.9 gear rat., 500 CPR encoder	3
Timing Pulley	0.375" belt width, 0.2" pitch, 0.891" pitch ⌀	3
Timing Belt	0.375" width, 0.2" pitch, neoprene, 2.1m length	3
Stainless Precision Shaft	1m length, 16mm ⌀	6
Linear Bearing	16 mm ID, 26mm OD	6
Radial Bearing (idler)	Double-sealed, 30mm OD, 10mm ID	3
Triangular Steel Plates	0.25" thick, water jet cut	2

5.3.2 Assemblies

All 3D printed parts were integrated with off-the-shelf components through either press fitting or metric bolts. The joints were produced using a similar design to the Rostock (as shown in figures 5.5a and 5.5b). This design used three separate 3D printed parts and three M5 bolts in each joint. Figures 5.6 through 5.9 show closeup photos of other design decisions implemented on the robot.



(a) U-joints used in "knee" joints (b) U-joints used in "ankle" joints

Figure 5.5: 3D printed universal joints used in the "ankle" and "knee joints" of the Delta robot

5.4 Software and Hardware

The TMS320F28335 eZdsp™ Development Kit from Texas Instruments [55] was used to handle all control decisions in the Delta. The board uses the TMS320F28335 (Delfino) microcontroller with 150 MHz cpu frequency. It comes with an on-board JTAG emulator, providing line-by-line debugging in Code Composer Studio™(CCS) development environment. This board was chosen for several reasons including: (1) large existing code base already developed, (2) great potential for future expansion, and (3) a convenient C-based coding environment.

The already existing code base originated with in-lab exercises for multiple courses taught in the College of Engineering Control Systems Lab (CO-

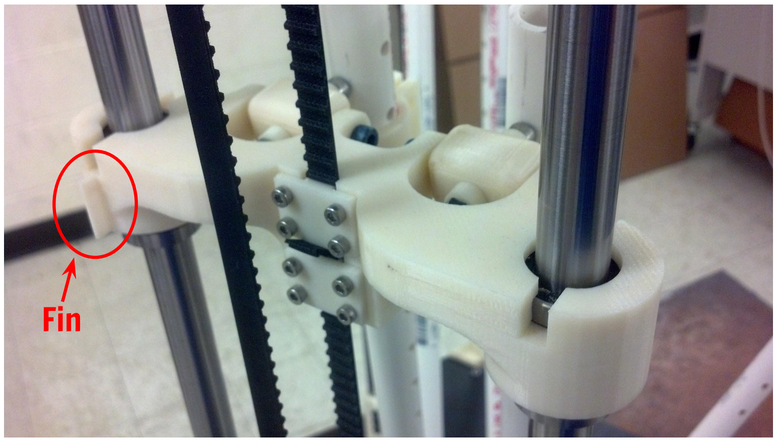


Figure 5.6: Closeup of one carriage. Two observations should be made: (1) the timing belt is exactly between the two linear guides and (2) the short fin on the left side of the carriage which is detected by the optointerrupter at the limits of the carriage trajectory.

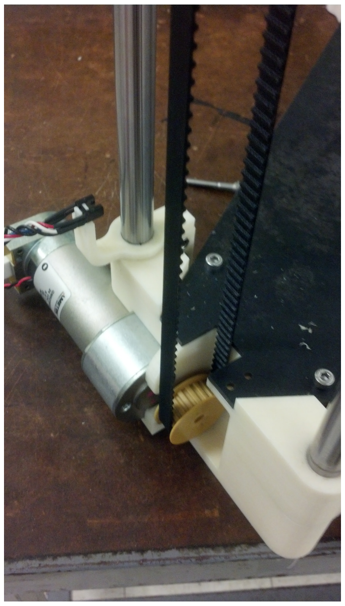


Figure 5.7: Closeup of the "bottom corner." Note the fixture on the left linear guide holding the optointerrupter. This fixture is a separate part, which can also easily be installed on at the upper limit of the carriage trajectory. (Currently we only are using the lower optointerrupter to zero carriage positions.)

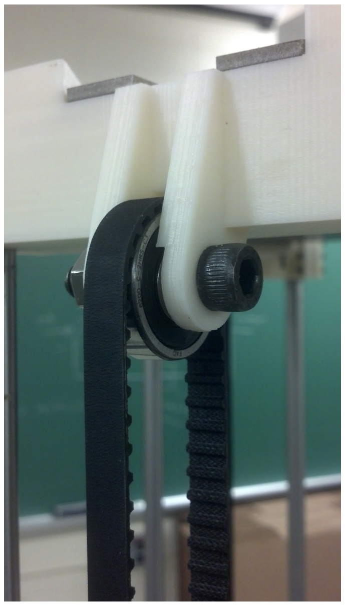


Figure 5.8: Closeup of the “top corner.” A radial bearing is used as an idler bearing for the timing belt. By tightening the M8 bolt, the 3D printed part clamps on the idler’s inner race.

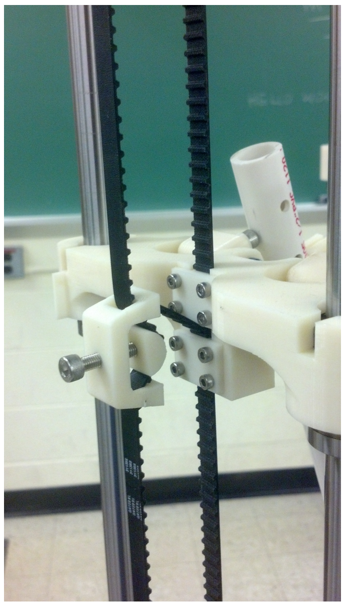


Figure 5.9: Closeup of the tensioner (used to manually tighten the timing belt before operation). There is a nut (not visible) keeping the M5 bolt from being pushed out by tension in the belt.

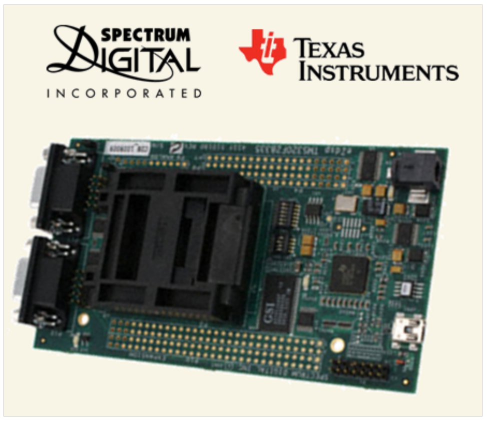


Figure 5.10: The TMS320F28335 ™ Development Kit [56]

ECSL). Also, the use of TI microprocessors (including the Delfino) in these lab-based courses gave me first hand experience developing on TI microprocessors using CCS. This experience included the use of the DSP operating system DSP BIOS, which gives the developer great control over on-device timing through a priority system used to manage hardware and software interrupts.

The large future potential of using this development kit comes from the breadth of input/output options available on the microcontroller. Among other features, the TMS320F28335 comes with 88 general purpose input/output pins, up to 18 pulse width modulated outputs, up to 16 ADC channels, and 2 built-in quadrature encoder interfaces.

The hardware interfaced with the microcontroller for the purposes of the current prototype includes: 3 DC gearmotors with encoders, 3 optical interrupters, 3 motor controllers, and a single external quadrature encoder interface to enable measurement of the rotations of the third DC gearmotor.

5.5 Future Plans

Immediate future plans include incorporation of continuously reconfigurable struts on the Delta Robot, and development of another manufacturing operation on the same Delta-based platform.

In research, this fabricated Delta robot provides a testbed for the reconfigurable design problem. As more capability is added to the existing system, the design approaches for reconfigurable systems can be tested and further developed. Approaches that work on paper can refined through experiments with a real system. For example, the case study performed in Chapter 2 could be strengthened using experimental testing to determine higher fidelity performance objectives grounded in data from the real systems.

In the larger picture, all of the progress made in the fabrication of a reconfigurable Delta robot has laid the groundwork for numerous future opportunities in both research and hands-on application— with future research in reconfigurable systems motivating further development of the fabricated design, and challenges in design fabrication motivating refinement of design approaches developed in research.

CHAPTER 6

CONCLUSION

The work in reconfigurable system design detailed a new method to design continuously reconfigurable systems for multiability. This method draws inspiration from the established product family design formulation given in Ref. [35]. It targets the simultaneous variable selection and optimization problem, in which the goal of a designer is to simultaneously choose which variables to make reconfigurable and the extent of their reconfigurability. My coauthor and I showed that this variable selection and ranging problem in the context of reconfigurability amounts to assigning value to subsets of the nonreconfigurable system's Pareto front in the performance space. In the case of systems designed for multiability through offline transformations, we exploit the fact that designers seek to optimize the reconfigurable product for m discrete tasks individually. We are thus able to focus on only the relevant points on each subset of the nonreconfigurable system's Pareto frontier— the designs which individually optimize each design objective. In doing so, we can distill each subset of the nonreconfigurable system's Pareto set into only m points. Provided a designer can specify the cost associated with different ranges of reconfigurability, we are able to construct an "extended Pareto front," which applies to the reconfigurable design problem. Each point on the extended Pareto front represents a fully reconfigurable design that is non-dominated in terms of individual task performance objectives and cost. Thus, for only one additional dimension in the performance space, designers can focus only on the non-dominated subsets of the nonreconfigurable system's Pareto front.

After detailing this method in reconfigurable system design, we applied it to a case study in the design of a reconfigurable Delta robot. In the case study, the Delta robot is meant to perform two tasks: a milling task, where end-effector rigidity is the performance objective, and a pick-and-place task, where the speed of the end effector is the performance objective. Using design metrics established in [37] to measure overall design rigidity, and original metrics to measure overal design speed, we framed the design problem using formulation 2.1. Next, we used a multiobjective genetic algorithm (MOGA) as a generating method to find points on or near the extended Pareto frontier of the reconfigurable design problem. This generation was performed for varying levels of reconfigurability cost.

After detailing a method in designing systems for reconfigurability, discussion shifted to the area of dynamic design, specifically, the integrated control and physical system design (co-design) of a simple compliant manipulator. After discussion of the necessary prerequisites (system model and co-design through direct transcription), I moved on to describe developing work in the co-design of a compliant member. GPOPS-II, a popular MATLAB-based software for pseudospectral methods (a special type of direct transcription) was used to perform numerical optimization for two separate formulations, one that sought to minimize time to perform a pick and place task, and another that sought to minimize energy consumption while performing a set task in a fixed amount of time. The minimum time formulation, which is a singular optimal control problem, proved to numerically difficult to solve due to vibrations imparted in the compliant member, believed to be a result of "bang-bang" style control. The second formulation that aimed to improve energy-efficiency met much more success. For parameter values given in Table 4.1, the codesign problem was solved successfully. After obtaining this solution, a parameter sweep of the member's in-plane thickness was performed, and the computation time required to solve the co-design problem was found to be highly dependent on the member's compliance.

After discussion of the research in codesign of a compliant member, I moved on to describe the detailed design and physical implementation of a reconfigurable Delta robot that was comprised of a mix of 3D printed and off-the shelf parts. I described the components used in the design, mechanical, hardware, and software, before discussing the future research applications of a Delta-robot-based multimanufacturing tool. This work in implementation brought together my research interests and desire to fabricate a real system, revealed important insights about the design problem, and laid the groundwork for future work in reconfigurable system design that straddles the gap between engineering design optimization in theory and application.

In this thesis, I have fulfilled the goals that I set forth at the onset of my Master's degree, submitting two (accepted) papers to reputable conferences, fabricating a reconfigurable Delta robot, and learning as much as possible in two short years. The projects presented in this document have pushed me to develop myself into a stronger researcher, writer, and designer. It is with these sharpened skills that I hope to further develop the research work detailed here into scientific inquiries that push the boundaries in the engineering design community.

APPENDIX A

ENERGY METHODS AND THE EULER-LAGRANGE EQUATION

Energy methods are an incredibly useful tool for deriving the equations of motion of mechanical systems. Unlike a Newtonian approach, which requires a force balance for each body in the system, energy methods do not require solving for reaction forces and allow dynamic equations to be derived using any valid set of coordinates. The dynamic equation determined using energy methods fits a particular form, the Euler-Lagrange equation.

We now derive the Euler-Lagrange equation for a mechanical system from the principle of virtual work. This derivation is fairly accessible method for a general audience, only requiring knowledge of physics and multivariable calculus.

The following proof is adapted from [51]. We derive the Euler-Lagrange equation for mechanical systems of n rigid bodies.

Consider a system of n rigid bodies that is at rest. For a static system in equilibrium, the net external force and external torque on each member of the system is zero. Mathematically, we can write:

$$\sum_{i=1}^{n} \mathbf{F}_i = 0 \tag{A.1}$$

Where $\mathbf{F}_i$ the net force on the ith component.

Next, consider applying an infinitesimal displacement to this system that is consistent with all kinematic constraints. Typically this small displacement is given the name *virtual* displacement, and the symbol $\delta\mathbf{x}$. We can state the infinitesimal amount of work done, δW by undergoing virtual displacement $\delta\mathbf{x}$ as:

$$\delta W = \sum_{i=1}^{n} \mathbf{F}_i \cdot \delta\mathbf{x}_i = \left(\sum_{i=1}^{n} \mathbf{F}_i\right) \cdot \delta\mathbf{x}_i \tag{A.2}$$

Note that virtual displacements are treated just like differentials, and the above expression is just a generalization of the scalar relation $dW = \mathbf{F} \cdot d\mathbf{x}$. From Equation A.1 we know that each term in this sum must be zero for any valid set of virtual displacements. Thus, for all admissible virtual displacements, the associated virtual work on the system must be zero. This is the principle of virtual work, and is written mathematically in equation A.3.

$$\delta W = \sum_{i=1}^{n} \mathbf{F}_i \cdot \delta \mathbf{x}_i = 0 \tag{A.3}$$

We can separate the forces that originate with constraints and rewrite Equation A.3 as:

$$\sum_{i=1}^{n} \mathbf{F}_i^{(a)} \cdot \delta \mathbf{x}_i + \sum_{i=1}^{n} \mathbf{f}_i^{(c)} \cdot \delta \mathbf{x}_i = 0, \tag{A.4}$$

where $\mathbf{F}_i^{(a)}$ and $\mathbf{f}_i^{(c)}$ are the total applied force and reaction forces on component i, respectively. In conservative systems, we can make the assumption that constraint forces act perpendicularly to valid virtual displacements. This means that the second term in Equation A.4 can be set equal to zero. Equation A.4 becomes:

$$\sum_{i=1}^{n} \mathbf{F}_i^{(a)} \cdot \delta \mathbf{x}_i = 0 \tag{A.5}$$

D'Alembert's principle allows us to restate equation A.5 just as correctly for dynamic systems provided we add a term for inertial forces. We can thus state the "dynamic version" of equation A.5 as:

$$\sum_{i=1}^{n} \left(\mathbf{F}_i^{(a)} - m_i \ddot{\mathbf{x}}_i \right) \cdot \delta \mathbf{x}_i = 0 \tag{A.6}$$

At this point, $\delta \mathbf{x}$ is not totally arbitrary. To make virtual displacements truly arbitrary, displacements need to be transformed into a coordinate system that implicitly satisfies kinematic constraints. If we can write equation A.6 in terms of virtual displacements in these generalized coordinates, dynamic equations for the system can be determined. Suppose we can express displacements $\mathbf{x}_i$ as functions of a generalized coordinate vector $(\mathbf{q})$ and time, i.e., if we have $\mathbf{x}_i = \mathbf{x}_i(q_1, ..., q_m, t)$, then we can rewrite the first term in Equation A.6 in terms of generalized virtual displacements δq as:

$$\sum_{i=1}^{n} \mathbf{F}_i^{(a)} \cdot \delta \mathbf{x}_i = 0 = \sum_{i=1}^{n} \mathbf{F}_i^{(a)} \cdot \left(\sum_{j=1}^{m} \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \delta q_j \right) \tag{A.7}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{F}_i^{(a)} \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \delta q_j = 0 \tag{A.8}$$

$$= \sum_{j=1}^{m} \sum_{i=1}^{n} \mathbf{F}_i^{(a)} \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \delta q_j = 0 \tag{A.9}$$

$$= \sum_{j=1}^{m} Q_j \delta q_j = 0 \tag{A.10}$$

Where $Q_j \triangleq \sum_{i=1}^{n} \mathbf{F}_i^{(a)} \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right)$, and is termed the jth "generalized force" on the system.

We can also rewrite the second term in Equation A.6 in terms of generalized virtual displacements as:

$$\sum_{i=1}^{n} m_i \ddot{\mathbf{x}}_i \cdot \delta \mathbf{x}_i = \sum_{i=1}^{n} m_i \ddot{\mathbf{x}}_i \cdot \left(\sum_{j=1}^{m} \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \delta q_j \right) \tag{A.11}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} m_i \ddot{\mathbf{x}}_i \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \delta q_j = 0 \tag{A.12}$$

$$= \sum_{j=1}^{m} \sum_{i=1}^{n} m_i \ddot{\mathbf{x}}_i \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \delta q_j = 0 \tag{A.13}$$

The term $\sum_{i=1}^{n} m_i \ddot{\mathbf{x}}_i \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right)$ in equation A.13 can be rewritten with the relation:

$$\sum_{i=1}^{n} m_i \ddot{\mathbf{x}}_i \cdot \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) = \sum_{i=1}^{n} \left\{ \frac{d}{dt} \left(m_i \dot{\mathbf{x}}_i \cdot \frac{\partial \mathbf{x}_i}{\partial q_j} \right) - m_i \dot{\mathbf{x}}_i \cdot \frac{d}{dt} \left(\frac{\partial \mathbf{x}_i}{\partial q_j} \right) \right\} \tag{A.14}$$

We make two substitutions into equation A.14, replacing the first and second terms in braces. To make these substitutions, we need to identify a few valid equalities that result from calculus-based relations. First, consider the relation which follows from the defined equality $\mathbf{x}_i = \mathbf{x}_i(q_1, ..., q_m, t)$, and the rules of partial differentiation:

$$\dot{\mathbf{x}}_i = \sum_{k=1}^{m} \frac{\partial \mathbf{x}_i}{\partial q_k}\dot{q}_k + \frac{\partial \mathbf{x}_i}{\partial t} \tag{A.15}$$

Next, rewrite the last term in equation A.14 using the rules of partial differentiation.

$$\frac{d}{dt}\left(\frac{\partial \mathbf{x}_i}{\partial q_j}\right) = \sum_{k=1}^{m} \frac{\partial^2 \mathbf{x}_i}{\partial q_j \partial q_k}\dot{q}_k + \frac{\partial^2 \mathbf{x}_i}{\partial q_j \partial t} \tag{A.16}$$

We can make the observation that:

$$\frac{d}{dt}\left(\frac{\partial \mathbf{x}_i}{\partial q_j}\right) = \frac{\partial \dot{\mathbf{x}}_i}{\partial q_j}, \tag{A.17}$$

where the last equality follows from the realization that the right hand side of Eqn. (A.16) is simply the partial derivative of the right hand side of Eqn. (A.15) with respect to q_j. Also from Eqn. (A.15), we get the following relation:

$$\frac{\partial \dot{\mathbf{x}}_i}{\partial \dot{q}_j} = \frac{\partial \mathbf{x}_i}{\partial q_j} \tag{A.18}$$

If we substitute Equations A.16 and A.17 into Equation A.14, we obtain:

$$\sum_{i=1}^{n} m_i \ddot{\mathbf{x}} \cdot \frac{\partial \mathbf{x}_i}{\partial q_j} = \sum_{i=1}^{n} \left\{ \frac{d}{dt}\left(m\dot{\mathbf{x}}_i \cdot \frac{\partial \dot{\mathbf{x}}}{\partial \dot{q}_j}\right) - m_i \dot{\mathbf{x}}_i \cdot \frac{\partial \dot{\mathbf{x}}_i}{\partial q_j} \right\} \tag{A.19}$$

We can substitute this relation into equation A.14 to obtain:

$$\sum_{j=1}^{m} \left\{ \frac{d}{dt}\left(\frac{\partial}{\partial \dot{q}_j} \left(\sum_{i=1}^{n} \frac{1}{2} m_i \dot{\mathbf{x}}_i^2 \right)\right) - \frac{\partial}{\partial q_j}\left(\sum_{i=1}^{n} m_i \dot{\mathbf{x}}_i^2 \right) \right\} \partial q_j \tag{A.20}$$

Combining Equations A.20 and A.10 we can write an equation in terms of the systems kinetic energy of the system, T.

$$\sum_{j=1}^{m} \left[\left\{ \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial T}{\partial q_j} \right\} - Q_j \right] \delta q_j = 0 \tag{A.21}$$

The portion of the generalized force Q_j which originates from conservative forces, can be written as the negative spacial gradient of potential energy, i.e. we can write $Q_{j,cons} = -\frac{\partial V}{\partial q_j}$. We can restate equation A.21 in terms of potential energy V, and generalized non-conservative forces, $Q_{j,nc}$ as:

$$\sum_{j=1}^{m} \left[\left\{ \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_j}\right) - \frac{\partial (T-V)}{\partial q_j} \right\} - Q_{j,nc} \right] \delta q_j = 0 \tag{A.22}$$

If potential energy V of the system does not depend directly on the derivative of the generalized coordinates, we can add the V term to the first derivative taken. This leaves:

$$\sum_{j=1}^{m} \left[\left\{ \frac{d}{dt}\left(\frac{\partial (T-V)}{\partial \dot{q}_j}\right) - \frac{\partial (T-V)}{\partial q_j} \right\} - Q_{j,nc} \right] \delta q_j = 0 \tag{A.23}$$

Recall that because the generalized displacements are independent, Equation A.23 must remain true for arbitrary virtual displacements. Thus, each term in square brackets must evaluate to zero for all q_j. This equality is the **Euler-Lagrange equation**:

$$\frac{d}{dt}\left(\frac{\partial (T-V)}{\partial \dot{q}_j}\right) - \frac{\partial (T-V)}{\partial q_j} - Q_{j,nc} = 0 \quad \forall q_j \tag{A.24}$$

Using the Euler-Lagrange equation to derive the dynamic equations of a system can be done reasonably simply. The procedure is:

1. Write coordinates of each body in terms of generalized system which implicitly satisfies constraints.

2. Write potential energy in terms of a truly independent coordinates. Any potential energy which satisfies $-\frac{\partial V}{\partial q_j} = Q_{j,cons}$ is valid.

3. For each generalized coordinate, write the Euler-Lagrange equation. This dynamic equation needs to be satisfied for any valid trajectory as a necessary condition.

APPENDIX B

DELTA ROBOT FORWARD AND INVERSE KINEMATICS

Overview

We now derive the forward and inverse kinematics for the Delta robot. One important point to understanding the below proofs is the fact that the kinematics of the Delta robot are found with the same equations as would be used to determine the intersection of three spheres where the center of sphere i is taken as the xyz coordinate of the ith carriage, offset by $\frac{D}{2\sqrt{3}}$ in the xy plane toward the center of the build envelope. Note that we define the distance l as the length of each strut, and the distance D is the side length of the triangular end effector. (see figure B.1).

From this point on, we consider the position of the end effector as the intersection of three spheres: $\mathbf{S_1}$, centered at (x_1, y_1, z_1); $\mathbf{S_2}$ centered at (x_2, y_2, z_2), and $\mathbf{S_3}$ centered at (x_3, y_3, z_3). We refer to the the equilateral triangle in the xy plane that results from connecting the xy coordinates of the three spheres as the "build envelope". We call the side length of this equilateral triangle L.

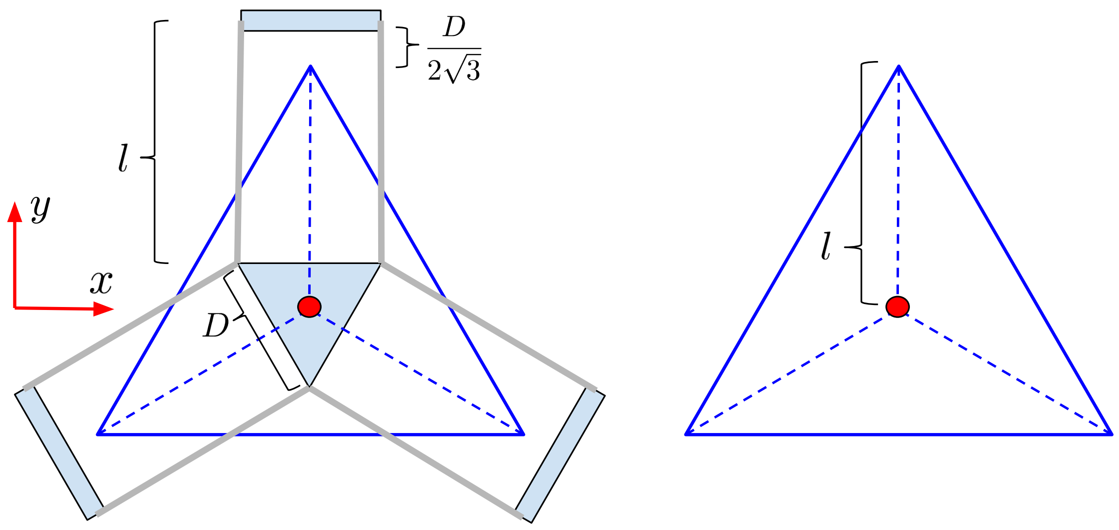


Figure B.1: Left: Top view of the Delta robot build area. Right: Equivalent kinematic representation.

Inverse Kinematics

Let (p, q, r) be any position of end effector within the build envelope. Let the origin of our xyz coordinate frame fall at $(p, q, r) = (0, 0, 0)$. Let (x_i, y_i, z_i) be the position of i^{th} carriage. Equations shown in B.1 follow easily the geometry of figure B.2.

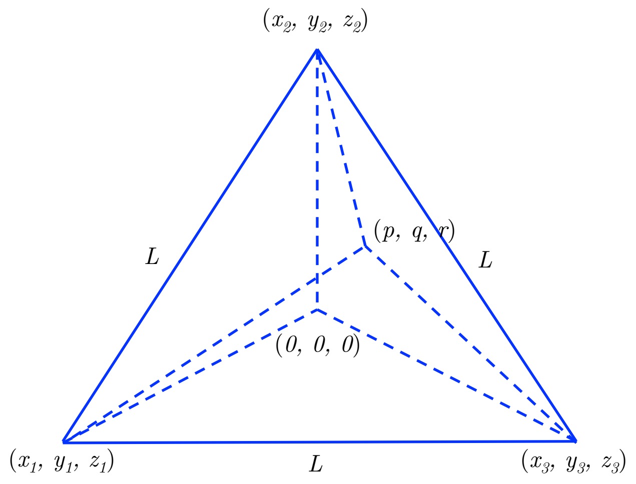


Figure B.2: Top view of the build envelope with struts attached to the end effector.

$$\begin{aligned}(x_1, y_1, z_1) &= (\frac{-L}{2}, \frac{-L}{2\sqrt{3}}, z_0)\\(x_2, y_2, z_2) &= (0, \frac{L}{\sqrt{3}}, z_0)\\(x_3, y_3, z_3) &= (\frac{L}{2}, \frac{-L}{2\sqrt{3}}, z_0)\end{aligned} \tag{B.1}$$

Equations for each of the three spheres ($\mathbf{S_1}, \mathbf{S_2}$ and $\mathbf{S_3}$) can be written as following:

$$\begin{aligned}\mathbf{S_1} &: \left(x + \frac{L}{2}\right)^2 + \left(y + \frac{L}{2\sqrt{3}}\right)^2 + (z - z_1)^2 = l^2\\\mathbf{S_2} &: x^2 + \left(y - \frac{L}{\sqrt{3}}\right)^2 + (z - z_2)^2 = l^2\\\mathbf{S_3} &: \left(x - \frac{L}{2}\right)^2 + \left(y + \frac{L}{2\sqrt{3}}\right)^2 + (z - z_3)^2 = l^2\end{aligned} \tag{B.2}$$

As the carriages move along z - axes only, x_i and y_i are constant for each of the carriages for any end effector location. Hence the **inverse kinematic** problem then reduces to finding the sphere positions (z_1, z_2, z_3) for the given end effector position (p,q,r). This can be achieved by substituing $(x,y,z) = (p,q,r)$ in Equation B.2 and rearraging the terms.

$$\begin{aligned} z_1 &= r + \sqrt{l^2 - \left(p + \frac{L}{2}\right)^2 - \left(q + \frac{L}{2\sqrt{3}}\right)^2} \\ z_2 &= r + \sqrt{l^2 - p^2 - \left(q - \frac{L}{\sqrt{3}}\right)^2} \\ z_3 &= r + \sqrt{l^2 - \left(p - \frac{L}{2}\right)^2 - \left(q + \frac{L}{2\sqrt{3}}\right)^2} \end{aligned} \tag{B.3}$$

Forward Kinematics

The **forward kinematic** problem is to find the position of end effector (p,q,r) for the given positions of carriages (z_1, z_2, z_3). We can do this by 'solving for' p, q and r, simultaneously in following equations.

$$\begin{aligned} &\left(p + \frac{L}{2}\right)^2 + \left(q + \frac{L}{2\sqrt{3}}\right)^2 + (r - z_1)^2 = l^2 \\ &p^2 + \left(q - \frac{L}{\sqrt{3}}\right)^2 + (r - z_2)^2 = l^2 \\ &\left(p - \frac{L}{2}\right)^2 + \left(q + \frac{L}{2\sqrt{3}}\right)^2 + (r - z_3)^2 = l^2 \end{aligned} \tag{B.4}$$

We can rewrite the set of equations B.4 more generally as:

$$(p - x_1)^2 + (q - y_1)^2 + (r - z_1)^2 = l^2 \tag{B.5}$$

$$(p - x_2)^2 + (q - y_2)^2 + (r - z_2)^2 = l^2 \tag{B.6}$$

$$(p - x_3)^2 + (q - y_3)^2 + (r - z_3)^2 = l^2 \tag{B.7}$$

Expanding, we obtain:

$$\begin{aligned} p^2 + q^2 + r^2 + x_1^2 + y_1^2 + z_1^2 - 2px_1 - 2qy_1 - 2rz_1 = l^2 \\ p^2 + q^2 + r^2 + x_2^2 + y_2^2 + z_2^2 - 2px_2 - 2qy_2 - 2rz_2 = l^2 \\ p^2 + q^2 + r^2 + x_3^2 + y_3^2 + z_3^2 - 2px_3 - 2qy_3 - 2rz_3 = l^2 \end{aligned} \tag{B.8}$$

Let, $w_i \triangleq x_i^2 + y_i^2 + z_i^2$

$$p^2 + q^2 + r^2 + w_1 - 2px_1 - 2qy_1 - 2rz_1 = l^2 \tag{B.9}$$
$$p^2 + q^2 + r^2 + w_2 - 2px_2 - 2qy_2 - 2rz_2 = l^2 \tag{B.10}$$
$$p^2 + q^2 + r^2 + w_3 - 2px_3 - 2qy_3 - 2rz_3 = l^2 \tag{B.11}$$

Subtract B.10 from B.9 and subtract B.11 from B.9. The result is two linear equations:

$$w_1 - w_2 - 2p(x_1 - x_2) - 2q(y_1 - y_2) - 2r(z_1 - z_2) = 0 \tag{B.12}$$
$$w_1 - w_3 - 2p(x_1 - x_3) - 2q(y_1 - y_3) - 2r(z_1 - z_3) = 0 \tag{B.13}$$

Solving B.12 for p and B.13 for q we obtain:

$$p = \frac{w_1 - w_2 - 2q(y_1 - y_2) - 2r(z_1 - z_2)}{2(x_1 - x_2)} \tag{B.14}$$
$$q = \frac{w_1 - w_3 - 2p(x_1 - x_3) - 2r(z_1 - z_3)}{2(y_1 - x_3)} \tag{B.15}$$

Substituting for p in equation B.13 using B.14, and substituting for equation q in equation B.12using B.15 leads to:

$$w_1 - w_3 = \frac{(w1 - w2 - 2q(y_1 - y_2) - 2r(z_1 - z_2))(x_1 - x_3)}{x_1 - x_2} + 2q(y_1 - y_3) + 2r(z_1 - z_3) \tag{B.16}$$

$$w_1 - w_2 = \frac{(w1 - w3 - 2p(x_1 - x_3) - 2r(z_1 - z_3))(y_1 - y_2)}{y_1 - y_3} + 2p(x_1 - x_2) + 2r(z_1 - z_2) \tag{B.17}$$

Solving equation B.16 for q gives q entirely as a function of r. Similarly, solving equation B.17 for p gives p entirely as a function of r.

Solving equation B.16 for q and equation B.17 for r gives:

$$q = \frac{w_1 - w_3)(x_1 - x_2) - (w_1 - w_2)(x_1 - x_3)}{2(y_1 - y_3)(x_1 - x_2) - 2(y_1 - y_2)(x_1 - x_3)} + \frac{(z_1 - z_2)(x_1 - x_3) - (z_1 - z_3)(x_1 - x_2)}{(y_1 - y_3)(x_1 - x_2) - (y_1 - y_2)(x_1 - x_3)} r \quad \text{(B.18)}$$

$$p = \frac{w_1 - w_2)(y_1 - y_3) - (w_1 - w_3)(y_1 - y_2)}{2(x_1 - x_2)(y_1 - y_3) - 2(x_1 - x_3)(y_1 - y_2)} + \frac{(z_1 - z_3)(y_1 - y_2) - (z_1 - z_2)(y_1 - y_3)}{(x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_2)} r \quad \text{(B.19)}$$

Equations B.19 and B.18 linear in z. From now on we represent them more compactly as:

$$p = a + br \quad \text{(B.20)}$$

$$q = c + dr \quad \text{(B.21)}$$

Where constants a, b, c, and d can be viewed in equations B.18 and B.19.

We can substitute for p and q in equation B.5. This leaves a quadratic equation purely in terms of r.

$$r^2(1 + d^2 + b^2) + r(2b(a - x_1) + 2d(c - y1) + 2z_1) + (a - x_1)^2 + (c - y_1)^2 + (z_1)^2 - l^2 = 0 \quad \text{(B.22)}$$

The roots of the above equation can be found using the quadratic formula. The roots of this equation will be either both real or both imaginary. If they are both imaginary, the three spheres do not intersect in 3d space. If the roots are real-valued, the lower root will give the z position of the end effector, and equations B.20 and B.21 can be used to solve for the x and y end effector positions, respectively.

REFERENCES

[1] A. Olewnik, T. Brauen, S. Ferguson, and K. Lewis, "A framework for flexible systems and its implementation in multiattribute decision making," *Journal of Mechanical Design, Transactions of the ASME*, vol. 126, no. 3, pp. 412–419, 2004.

[2] A. Siddiqi, O. de Weck, and K. Iagnemma, "Reconfigurability in planetary surface vehicles: Modelling approaches and case study," *JBIS - Journal of the British Interplanetary Society*, vol. 59, no. 12, pp. 450–460, 2006.

[3] J. L. Cohon, *Multiobjective Programming and Planning.* Dover Publications Incorporated, 2003.

[4] J. P. Ignizio, "Generalized goal programming: An overview," *Computers and Operations Research*, vol. 10, no. 4, pp. 277–289, 1983.

[5] J. Koski, "Defectiveness of weighting method in multicriterion optimization of structures," *Communications in Applied Numerical Methods*, vol. 1, pp. 333–337, 1985.

[6] J. D. Indraneel Das, "Normal-boundary intersection: An alternative method for generating pareto optimal points in multicriteria optimization problems," NASA Langley Research Center, Tech. Rep., 1996.

[7] J. D. I. Das, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimizaiton problems," *SIAM Journal of Optimization*, vol. 8, pp. 631–657, 1998.

[8] A. Messac, A. Ismail-Yahaya, and C. Mattson, "The normalized normal constraint method for generating the pareto frontier," *Structural and Multidisciplinary Optimization*, vol. 25, pp. 86–98, 2003.

[9] O. d. W. O. I.Y. Kim, "Adapt weighted sum method for multiobjective optimization," in *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, September 2004.

[10] S. Marglin, *Public Investment Criteria. Studies in the Economic Development of India, London, and Cambridge.* Cambridge, MA: Allen and Unwin and MIT Press, 1967.

[11] M. Zeleny, "Compromise programming," in *Multiple Criteria Decision Making*, J. Cochrane and M. Zeleny, Eds. Columbia: University of South Carolina Press, 1973, pp. 262–301.

[12] M. J. Rentmeesters, W. K. Tsai, and K.-J. Lin, "Polybot: A modular reconfigurable robot," in *Proceedings from the Second IEEE International Conference on Engineering of Complex Computer Systems*, 1996.

[13] H. Simon, *Models of Man: Social and Rational.* John Wiley and Sons, Inc., 1957.

[14] D. Jones and M. Tamiz, *Practical Goal Programming*, F. S. Hillier, Ed. Springer, 2010.

[15] E. J. Cramer, J. E. Dennis, P. D. Frank, R. M. Lewis, and G. R. Shubin, "Problem formulation for multidisciplinary optimization," in *AIAA Symposium on Multidisciplinary Design Optimization*, 1993.

[16] C. McAllister, T. Simpson, and M. Yukish, "Goal programming aapplication in multidisciplinary design optimization," in *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2000.

[17] J. Holland, *Adaptation in Natural and Artificial Systems.* MIT Press, 1975.

[18] D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley Publishing Company, 1989.

[19] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*, S. Ross and R. Weber, Eds. John Wiley and Sons, 2001.

[20] C. Fonseca and P. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 416–423.

[21] A. Siddiqi and O. de Weck, "Modeling methods and conceptual design principles for reconfigurable systems," *Journal of Mechanical Design, Transactions of the ASME*, vol. 130, no. 10, pp. 1 011 021–10 110 215, 2008.

[22] R. Khire and A. Messac, "Selection-integrated optimization (sio) methodology for optimal design of adaptive systems," *Journal of Mechanical Design, Transactions of the ASME*, vol. 130, no. 10, pp. 1 014 011–10 140 113, 2008.

[23] S. Ferguson, K. Lewis, A. Siddiqi, and O. de Weck, “Flexible and reconfigurable systems: Nomenclature and review,” in *Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, vol. 6, 2007, pp. 249–263.

[24] Y. Koren, “The rapid responsiveness of rms,” *International Journal of Production Research*, vol. 51, no. 23-24, pp. 6817–6827, 2013.

[25] M. Mehrabi, A. Ulsoy, and Y. Koren, “Reconfigurable manufacturing systems: Key to future manufacturing,” *Journal of Intelligent Manufacturing*, vol. 11, pp. 403–419, 2000.

[26] S. Barbarino, O. Bilgen, R. Ajaj, M. Friswell, and D. Inman, “A review of morphing aircraft,” *Journal of Intelligent Material Systems and Structures*, vol. 22, no. 9, pp. 823–877, 2011.

[27] M. Yim, D. G. Duff, and K. Roufas, “Polybot: A modular reconfigurable robot,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[28] C. A. Mattson and A. Messac, “Concept selection using s-pareto frontiers,” *AIAA Journal*, vol. 41, pp. 1190–1198, 2003.

[29] B. Literman, P. Cormier, and K. Lewis, “Concept analysis for reconfigurable products,” in *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computer Information in Engineering Conference*, vol. 3, no. PARTS A AND B, 2012, pp. 209–222.

[30] T. Simpson, J. Maier, and F. Mistree, *The Information Revolution: Present and Future.* Greenwich, Connecticut: Ablex Publications, 1998, ch. Mass Customization in the Age of Information: The Case for Open Engineering Systems.

[31] M. D. Patterson, D. J. Pate, and B. J. German, “Performance flexibility of a reconfigurable family of uavs,” in *AIAA Aviation, Technology, Integration and Operations (ATIO) Conference*, 2011.

[32] S. Ferguson and K. Lewis, “Investigating the interaction between reconfigurability and system mass using multidisciplinary design optimization,” in *AIAA/ASME/ASCE/AHS/ACE Structures, Structural Dynamics, and Materials Conference*, 2008.

[33] S. Ferguson, E. Kasprzak, and K. Lewis, “Designing a family of reconfigurable vehicles using multilevel multidisciplinary design optimization,” *Structural and Multidisciplinary Optimization*, vol. 39, no. 2, pp. 171–186, 2009.

[34] T. W. Simpson, "Product platform design and customization: Status and promise," *Artivicial Intelligence for ENgineering Design, Analysis and Manufacturing*, vol. 18, pp. 3–20, 2004.

[35] R. Fellini, M. Kokkolaras, P. Papalambros, and A. Perez-Duarte, "Platform selection under performance bounds in optimal design of product families," *Journal of Mechanical Design*, vol. 127, pp. 524–535, 2005.

[36] L. Rey and R. Clavel, *The Delta Parallel Robot.* Springer, 1999, ch. 29, pp. 401–417.

[37] E. Courteille, D. Deblaise, and P. Maurine, "Design optimization of a delta-like parallel robot through global stiffness performance evaluation," in *International Conference on Intelligent Robots and Systems*, 2009, pp. 5159–5166.

[38] H. Lipkin and T. Patterson, "Geometrical properties of modelled robot elasticity: Part i - decomposition," in *22nd Biennial Mechanisms Conference*, vol. 45, 1992, pp. 179–185.

[39] G. R. Eisler, R. Robinett, D. Segalman, , and J. Feddema, "Approximate optimal trajectories for flexible-link manipulator slewing using recursive quadratic programming," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 115, no. 3, pp. 405–410, 1993.

[40] D. Liberzon, *Calculus of Variations and Optimal Control Theory: A Concise Introduction.* Princeton University Press, 2012.

[41] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming.* Society for Industrial and Applied Mathematics, 2010, ch. 4, pp. 124–218.

[42] J. T. Allison, T. Guo, and Z. Han, "Co-design of an active suspension using simultaneous dynamic optimization," *To Appear in Journal of Mechanical Design*, DOI: 10.1115/1.4027335.

[43] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes.* Society for Industrial and Applied Mathematics and the Mathematical Optimization Society, 2010, ch. 10, pp. 287–324.

[44] K. Kozak, I. Ebert-Uphoff, and W. Singhose, "A fast robot with parallel geometrylocally linearized dynamic analysis of parallel manipulators and application of input shaping to reduce vibrations," *Journal of Mechanical Design*, vol. 126, pp. 156–168, 2004.

[45] F. Pierrot, A. Fournier, and P. Dauchez, "Towards a fully-parallel 6 dof robotbot for high-speed applications," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, 1991, pp. 1288–1293.

[46] K. M. Lee and D. K. Shah, "Dynamic analysis of a three-degrees-of-freedom in-parallel actuated manipulator," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 1288–1293, 1991.

[47] Y. Li, J. Wang, L.Wang, and X. Liu, "Inverse dynamics and simulation of a 3-dof spacial parallel manipulator," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[48] J. Allison and D. R. Herber, "Multidisciplinary design optimization for dynamic engineering systems," *AIAA Journal, Special Issue on Multidisciplinary Design Optimization*, vol. 52, pp. 691–710, 2014.

[49] D. R. Herber, "Dynamic system design optimization of wave energy converters utilizing direct transcription," M.S. thesis, The University of Illinois at Urbana-Champaign, 2014.

[50] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quatrature collocation methods and sparse nonlinear programming," University of Florida, Gainesville, Tech. Rep., 2013.

[51] H. Goldstein, *Classical Mechanics.* Addison-Wesley Publishing Company, 1953.

[52] S. F. P. Saramago and V. Steffen, "Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system," *Mechanical Machine Theory*, vol. 33, pp. 883–894, 1998.

[53] R. R. dos Santos, V. Steffen, and S. de Fatima Pereira Saramago, "Optimal task placement of a serial manipulator for manipulability and mechanical power optimization," *Intelligent Information Management*, vol. 2, pp. 512–525, 2010.

[54] [Online]. Available: http://reprap.org/wiki/Rostock

[55] [Online]. Available: http://www.ti.com/product/tms320f28335

[56] [Online]. Available: http://www.element14.com/community/docs/DOC-39394/l/spectrum-digital-ti-tms320f28-based-ezdsp-f28335-starter-kit