An Interactive Computer Tool for Imprecise Calculations in Engineering Systems

by

Joseph E. Chen

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

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1995

Author .....
Department of Mechanical Engineering
December 21, 1994

Certified by .....
Kevin N. Otto
Assistant Professor of Mechanical Engineering
Thesis Supervisor

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

An Interactive Computer Tool for Imprecise Calculations in Engineering Systems

by

Joseph E. Chen

Submitted to the Department of Mechanical Engineering
on December 20, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Preliminary design is characterized by *imprecision*: the designer's uncertainty in choosing among alternatives. For any computation to occur, these informal descriptions must be translated into formal model representations, usually in the form of constraints. Conventional constraint based CAD systems are used to manipulate input and output variables, by allowing a user to adjust the variables' crisp values. The different variable values are iteratively specified and relaxed until a final configuration of variable values is accepted. ICPT, the Imprecise Constraint Propagation Tool, is developed to propagate a *set* of possible values which satisfy a given constraint network. In contrast to crisp constraint based CAD with which over-constrained systems of relations must be relaxed by the user, ICPT allows calculations to be made by converting crisp constraints into imprecise ones by *imprecise slackening*; it also finds a complete set of values which simultaneously satisfy all of the constraints. This global set-based approach of ICPT avoids running into infeasible space, and thus allows much of the iterative user specifications and exploration of design space to be made by the computing platform, reducing the iterative tasks of the user.

Thesis Supervisor: Kevin N. Otto
Title: Assistant Professor of Mechanical Engineering

Acknowledgments

I would like to thank my thesis advisor Kevin N. Otto for his support and guidance during this project, from choosing the right computer softwares to meeting publication deadlines. I also wish to thank him for the precious academic freedom he fosters in our Design Research Laboratory. Without this freedom, my pursuit of a wide range of research interests would be impossible.

I wish to thank my parents, Cong Long Chen and Xue Ping Zhou, for their loving support and care. Without their love, their examples of hard work and the values they taught me, this thesis, as well as a lot of other things in my life, would never have been possible.

I also wish to thank all my fellow students, friends and roommates. Their friendship makes my life at MIT and Ashdown House enjoyable and enlightening. Thank you: Steve, Ming, Tom, Rebecca, Kathy, Annette, Bruce, Torben, Jon, Earl, Sheh, Wallace, MingZheng, Brett, Q, José, Chris, Marc, Rama, Tony and Hyun Joon.

Finally, my deepest thanks to my fiancée Jing, for her endless love and patience. Jing, you are my consciousness and inspiration. I've never lost faith in you all these years across the ocean from you. I cherish your love and love you with all my heart.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Preliminary design is characterized by *imprecision*: the designer's uncertainty in choosing among alternatives. The method of imprecision [34, 66] is a formal theory that includes imprecision in design calculations. Recent application of it includes cost calculation using the Engine Development Cost Estimator provided by General Electric Aircraft Engines [23], computer systems design [28] and image enhancement [3].

For any computation to occur, these informal descriptions must be translated into formal model representations, usually in the form of constraints. Conventional constraint-based CAD systems are used to manipulate input and output variables, by allowing a user to adjust the variables' crisp values. The different variable values are iteratively specified and relaxed until a final configuration of variable values is accepted. This thesis develops an imprecisely constrained CAD tool to propagate a *set* of possible values which satisfy a given constraint network. An *imprecision transformation* is defined to induce imprecise specifications from specified variables to unspecified variables, either of which can be of the independent input or dependent output type. When the imprecise specifications are placed on the dependent variables exclusively, the transformation reduces to composition. When the imprecise specifications are placed on the input variables exclusively, the transformation reduces to Zadeh's extension principle used in fuzzy set mathematics. In the use of crisp constraint-based CAD, over-constrained systems of relations must be relaxed by the user. With an over-constrained system, however, it is shown that imprecise

constraints allow calculations to be made: the values which simultaneously satisfy all of the imprecise constraints can be calculated. Thus, using imprecise quantities in constraint-based CAD systems allows much of the iterative user specifications to be calculated instead by the computing platform, reducing the iterative tasks of the user.

1.1 Goals and Motivation

The goal of this work is to provide a CAD environment for imprecise constraint propagation, multi way. I consider an important class of problems defined by the constraints being a set of explicit or implicit nonlinear equations or inequalities. When designing with such systems, users often wish to adjust the values of some variables and to observe the corresponding changes on the remaining variables, while simultaneously keeping all of the constraints satisfied. Contrasted with conventional constraint-based tools, this work utilizes a set-based approach, and it provides additional set structures for choosing among alternatives.

One aim of this thesis is to extend the use of imprecisely constrained systems into the domain of *set-based concurrent engineering* [58]. When using a crisp constraint-based CAD tool, the system of relations and crisp variable values must be specified by the user along the progression of the design process. This can typically force decisions early in the process. Current industrial trends are away from this practice and instead toward application of concurrent engineering, which brings downstream and upstream decision-making to interact with each other. To implement this concept into engineering calculations and tools, a set-based approach to refining possibilities is needed. There are industrial justifications for this view. For example, Toyota's successful product development approach has been attributed to a set-based methodology [58]. In set-based concurrent engineering, sets of possible designs are communicated and modified among members of a cross-functional product development team. The preference functions are redefined and put into a common metric (such as monetary cost) to enable trade-off decisions.

1.2 Related Work

Design impression, first advocated by Wood and Antonsson [59, 62, 60, 61, 63, 64, 65], was later developed by Otto [32, 33, 35, 37, 38, 39, 42, 44, 45] into into a framework of formal design methodology [34]. For recent applications see [6, 7, 23, 43].

Some related work has been done by Diaz [10, 11] and Rao [49, 50], who consider optimizing imprecise engineering systems. This work, instead, is about presenting a user, not with a solution to an imprecise problem, but rather with the effects of different imprecise constraints on other variables. Sakawa and Yano [51, 52, 53, 54] discuss fuzzy multi-objective optimization. An appraisal of the use of fuzzy sets in optimization in general is given by Luhandjula [25].

The main interest of this thesis is to develop an interactive computer tool based on the method of imprecision, which has advantages over crisp constraint-based tools. Work has been done developing the propagation of crisp values through engineering models [1, 29, 55]. The use of such systems can be thought of as a network of variables. When enough variables are specified in a relation to leave a single variable remaining unspecified, such systems calculate that remaining variable value. This demonstrates what the unspecified value must be for the system to remain consistent. A computational spreadsheet for this purpose has recently been developed and presented by Ramaswamy and Ulrich [48].

I seek to develop the same constraint-propagation system, but to allow a user to specify imprecisely the values in the model, rather than being forced to choose exact, crisp values. The reason for doing so is that this will allow a user to observe the propagation of entire ranges of values, rather than only single ones. This places more of the computational burden on the computing platform, and less on the user, by reducing the need to make many adjustments to the variable values to gain insight into the proposed design solution. In this thesis, we will develop the extension principle for constraint systems, and demonstrate its usage and simplifications for various problems.

1.3 Organization of Thesis

The thesis organization is outlined in this section. Chapter 2 reviews the method of imprecision. A simple and efficient constrained interpolation scheme [7] for computer representation of preference curves is presented in this chapter. Chapter 3 develops a method for propagating imprecise constraints in engineering design. To enable interactivity, an Extension of Level Interval Algorithm is developed to handle arbitrary nonlinear constraints. The conditions for propagation are also discussed. Chapter 4 introduces features of the Imprecise Constraint Propagation Tool (ICPT), an interactive computer tool that propagates imprecise constraints. Chapter 5 shows two engineering design examples tested on ICPT. Design scenarios are depicted. In Chapter 6 future development of this work is outlined and discussed. In Chapter 7 the thesis is outlined. This work can be adapted into computer aided concurrent engineering and robust design, given its set-based approach and the additional set structure to perform evaluations. A user interface reference section is provided in the Appendix.

Chapter 2

Design Imprecision

2.1 Introduction

Preliminary design is characterized by *imprecision*: the designer's uncertainty in choosing among alternatives. The method of imprecision [34, 66], a formal theory that includes imprecision in design calculations, is reviewed in this chapter. First, a formal engineering model is constructed, then a suitable *trade-off strategy* is chosen to maximize performance. To propagate imprecise understanding through engineering tools, the preference must be constructed. In this chapter, a *constrained interpolation* scheme is developed for fitting a preference function to a finite number of known preference values.

2.2 Formal Models of Engineering Design

In a design process, a design is developed to satisfy the needs of a customer. In modern design practice, the needs of customers are first transformed into engineering terms. House of Quality [18] is a tool which helps product teams to realize customer needs through engineering functions and forms. Unlike engineering terms, customer needs are typically expressed using informal descriptions [2, 4, 24, 46] which can have many interpretations. In the modern product development environment, intense competitions and the opportunities offered by the rapid progress of information technology,

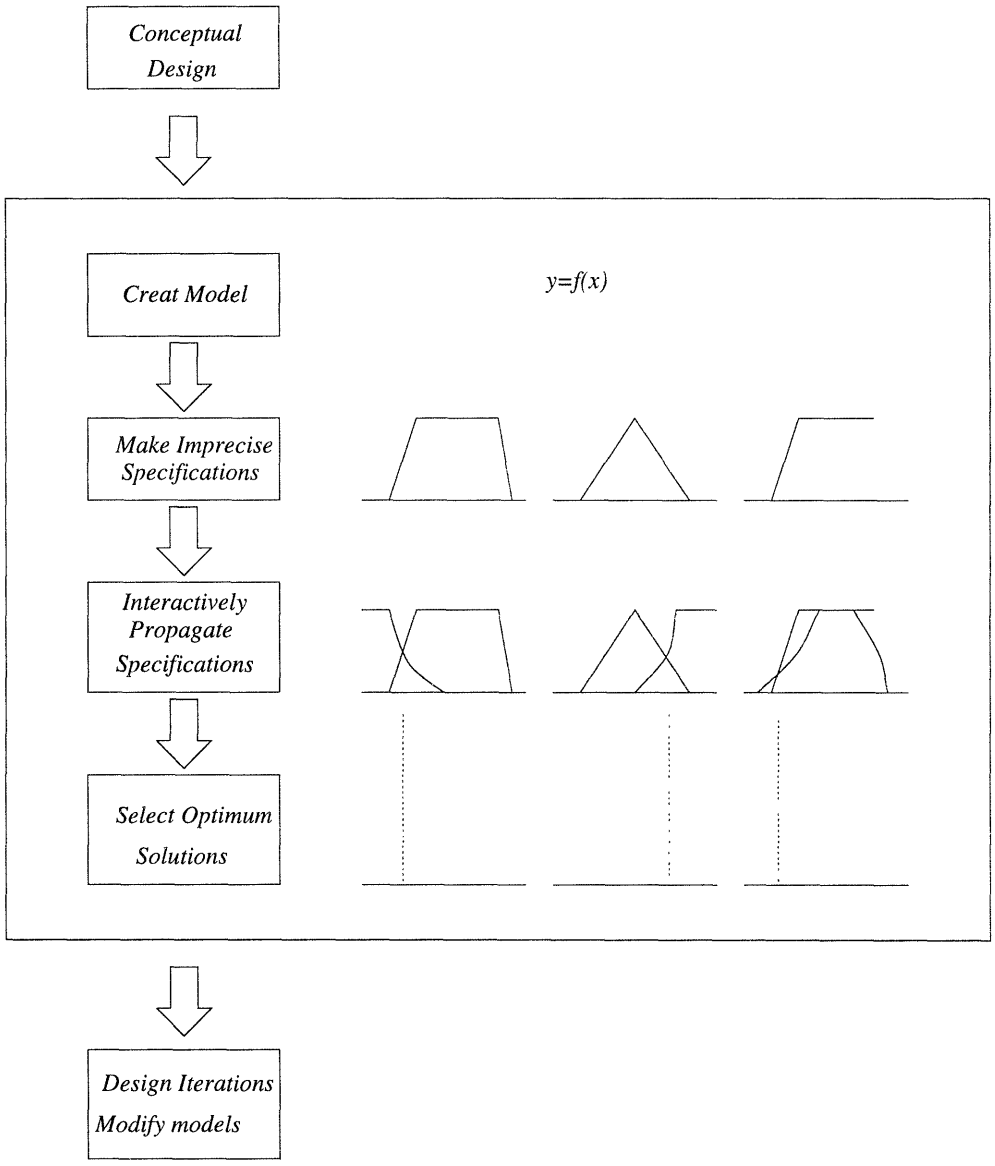

Figure 2-1: Design paradigm.

computers are used extensively in the design process. However, for any computation to occur, customers' informal descriptions must be translated into formal model representations. For example, design optimization methods require the translation of informal requirements into formal objectives and constraints. Concepts satisfying these objectives and constraints will be compared and improved through an iterative process. With tools like Pugh's concept selection chart [47], designers compare different aspects of a product with a "datum" and select the best concept. After a concept is selected, it must be formalized into a domain over which an optimization process can occur, and the formal constraints can be satisfied. This design paradigm is depicted in Figure 2-1. After conceptual design, an engineering model is created, with relations of engineering variables established. Imprecise and possible ranges on variables are then specified and propagated in the constraint network. This allows the designer to explore rapidly the design space and to come up with a set of solutions, among which an "optimum" solution exists, given all the preferences and constraints. These steps are covered in more detail in the next subsections.

2.2.1 Formal Structure

The assumptions underlying the construction of a formal model for a product are thoroughly studied by Otto [34]. It is assumed that all relevant aspects of a product can be quantified and related to customer requirements. Otto [34] defines the *design variable space* or DVS as "the set of considered possible alternative configurations, described using *design variables*, over which the designer has direct choice. Design variables are denoted d_i, $i = 1, ..., n$. The whole set of design variables is an n vector, $\vec{d}$. The space of $\vec{d}$ is denoted X and the set of valid values for d_i is denoted as X_i."

The *performance variable space* or PVS is "the dependent set of evaluated performances determined at each point in the DVS, described using *performance variables*. For each performance variable p_j, $j = 1, ..., q$, there is a mapping f_j such that $p_j = f_j(\vec{d})$. The set of performance variables is a q vector, $\vec{p} = \vec{f}(\vec{d})$. The subset of valid performance variable values Y is mapped from X and the set of valid values for p_j is denoted Y_j" [34].

Given this characterization of engineering design processes, the complete formal model assumed can be defined as "an *engineering model* consists of a DVS, a PVS, and a (possibly trivial) NVS (noise variable space)."

In this thesis a trivial NVS is assumed, except for Chapter 6, in which I consider stochastic noises. A formal model is usually a set of equations relating performance variables to design variables, but it could also be a computer program, a set of heuristic rules relating these variables, etc.

2.2.2 Design Imprecision and Objective Preferences

Once a formal model is created, the designer then wants to choose a set of variables that satisfy the constraints and in addition, to find the "best" if it is possible. The problem is in the preliminary design stage; there are no well-defined optimization goals, and even the feasibility of the model could be uncertain. The designer does not know what design variable values should be used in the model. In this thesis, this *imprecision* will be explicitly modeled by constructing a map from the variables into $[0,1] \subset \mathbb{R}$ indicating the satisfaction of the designer for values, and will be formally called the *designer preference* for values.

Otto defines [34] *preference* as "a map μ_k from a space X_k to $[0,1] \subset \mathbb{R}$,

$$\mu_k : X_k \rightarrow [0,1]$$

that preserves the designer's preferential order over X_k."

The rationale behind the preference could be other ambiguous constraints and considerations such as the availability of certain sizes of beams and screws, ergonomics, costs, supplier manufacturing capacities, etc. In the preliminary design stage, only those dominant and stringent constraints are considered. Less strict constraints and design intents are reflected in the designer preferences. The "best" design will emerge if all the constraints are satisfied in the design model and the preferences are maximized.

2.2.3 Propagating Preference in Design Model

To propagate preferences in design model one must deploy a trade-off strategy [37]. One must know if one configuration of design variables is better than another configuration. In this thesis a *non-compensating* strategy is used. So to rank two configurations, one looks at the worst preferences among all design variables and ranks them according to that. Preferences are propagated to a variable by achieving maximum *induced preference* [44] on that variable. Details of propagating preferences are discussed in Chapter 3.

2.3 Modeling Imprecision

2.3.1 Introduction

Imprecision in specification is a natural phenomenon which arises with any human conceptualization activity. In our particular interest, imprecision is an intrinsic aspect of *product development*, the process of understanding a customer's need and, based upon this, generating and embodying a product which can satisfy this need. Of particular interest to us is imprecision as it occurs in *engineering design*, which involves developing a product using engineering models and analysis.

Imprecision and vagueness are intrinsic aspects of engineering design. If (at the start of a design process) a proposed solution were neither imprecise nor vague, its description would be precise and it would therefore be a completed design. While stochastic uncertainty typically remains in a completed design description (e.g., dimensional tolerances), the nominal desired dimensions are precise. However, much of the early description of a design concept (physical dimensions, material properties, etc.) is vague and imprecise. At the early stages, a design team simply is not sure what values to use. Engineering calculations become difficult, because values for variables are unknown.

In this frame, imprecise mathematics becomes useful to represent the possibilities. Zadeh's extension principle, for example, can be used to propagate imprecise

understanding through relevant calculations and models. A comprehensive review of modeling imprecision and uncertainty in engineering design is given in [42]. Further work can be found in [36, 41, 44, 45, 62, 67].

For imprecision to be used during a design process, it must be represented. This means that a designer's understanding of the usefulness of values in a model must be represented, say, with fuzzy numbers. A well defined procedure for specifying the fuzzy numbers must be given, however, for design engineers to manipulate their models in an imprecise manner. One approach is to ask the designers and customers their preferences for various aspects of the design. They indicate their rank of preference (on a scale from 0 to 1) for every value of each variable, describing the design and its performance and constraints.

The preference information is usually (but not always) a convex fuzzy number. One can use the mathematics of fuzzy sets to operate on these imprecise descriptions of the design. For example, one can use preference data in conjunction with the usual engineering computations encountered in design. One can expand the process to map not just single values, but the entire imprecise set of variables specifying a design, such as geometric lengths. Preferences can also be mapped through fuzzy calculations to dependent variables such as material stresses, dynamic responses, or costs. Further, dependent variable preferences can also be back-mapped onto the variables specifying a design. This provides the designer an ability to manipulate the imprecise aspects of a design in an understandable and rapid way. The reader is referred to [42, 44, 45, 62, 67].

A two-fold problem arises. The first is to provide a well defined method for a design engineer to specify the preference values. The second problem is manipulating variables with an uncountable number of points, such as a common real-valued variable. A design engineer can only specify the preference for a finite subset of the points, and the remaining must have their preference determined through some form of interpolation. This thesis will present a method I have found useful for specifying a complete preference function across real valued variables. Methods are developed in this thesis to elicit preference values on a finite subset and to interpolate the re-

maining.

The interpolation problem is not solvable by a simple least-squares or spline method. Preference functions have constraints which common interpolation schemes do not satisfy. In particular, a preference function is bounded in $[0, 1]$. A *constrained* interpolation is required. I present a simple, efficient method to calculate a properly constrained preference function to a finite set of specified preference values.

The next section will present a useful method to determine preference values at a finite set of points. Section 3 presents the interpolation method and its proofs for fitting a preference function to these points. Section 4 will then discuss the relevance of this method for engineering calculations.

2.3.2 Measurement Theory

Measurement theory [21] provides a mathematically axiomatic method to construct preference values as used in this work. A comprehensive presentation of using measurement theory to construct evaluations in design is given in [32]. To construct a preference function for a variable, a set of values of the variable (here $\mathbb{R}$) must be known, and a reason for making the preference specifications. If a design engineer supplies a preference value of 1.0 to a variable value x_1 but a preference value of 0.0 to a different variable value x_0, there must be a reason for the difference between x_1 and x_0. A design engineer must have a reason for making this distinction. This informal reason behind the difference in formal preference I will denote by f, which is not a function, but merely a label attached to the informal reason.

Once a design engineer has a set of values X and has determined a reason f for specifying preference values, a preference function μ can be constructed over X using measurement theory [21, 32]. In particular, an interval scale construction will allow a designer to form a real valued scale μ which reflects the informal objective f. The designer must first identify which points in X have the least and most amount of the objective f, denoted x_{worst} and x_{best} respectively. These are designated with the amounts zero and one on the preference scale being constructed. Then for each other element $x_i \in X$, the designer must answer:

"On a scale of zero to one, what is your belief μ that you are indifferent between:

1) receiving the objective performance provided by x_i,

or

2) receiving the objective performance provided by x_{best} with certainty μ and receiving the objective performance provided by x_{worst} with certainty $(1-\mu)$?"

This constructs a real valued (measurable) preference function $\mu : X \rightarrow [0,1]$ which directly preserves the partial ordering (based on f) of the elements of X, and also the relative separation. A difference of 0.5 compared to 0.3 in μ means a larger difference in the informal reason f for constructing the preference function.

Historically this method has been associated with subjective probability. I maintain that for engineering design purposes, this method is suitable for constructing preference functions for propagating imprecision through engineering systems. In any case, for the interpolation algorithms presented below, how the finite subset of preference values is determined is irrelevant. A resulting subset of pairs $\{(x_1, \mu_1), \ldots, (x_n, \mu_n)\}$ is all that is required.

This basic measurement approach just given obviously assumes finite sets. On uncountable sets, further assumptions are required. The designer can only provide answers on a finite subset, which must then be interpolated. The next section will present the problem formulation and an efficient solution.

2.3.3 Interpolating Over Real Valued Variables

Preference functions have constraints which conventional least-squares and spline methods do not satisfy. The first constraint is that preference functions are usually monotonic and convex. A second constraint is that preference functions are bounded in $[0,1]$. These constraints will be detailed.

A preference function is usually monotonic and convex, defined as a *fuzzy number*

using the fuzzy-convex property:

$$\mu(\lambda x_i + (1-\lambda)x_j) \geq \min\{\mu(x_i), \mu(x_j)\}$$

where $\lambda \in [0,1]$ and $x_i, x_j \in \mathbb{R}$. Least squares and simple splines may not preserve the monotonicity and convexity. For example, consider a set of elicited preference values as shown in Table 2.1 for a set of material stress values. The data is fuzzy-convex. However, as shown in Figure 2-2, when using least squares or cubic splines to interpolate, the result is not fuzzy-convex.

Another constraint that any proposed interpolation scheme must address is to keep the preference function bounded within $[0,1]$. Again least squares and simple splines do not guarantee this boundary condition. For example, if two known preference values are closely spaced in the domain but widely separated in μ, then overshoots tend to occur, which may force the interpolation below 0 or above 1. In Figure 2-2, both interpolations exhibit preference-overshoot pathologies.

To preserve the local monotonicity and convexity, more advanced interpolation schemes were investigated beyond sample least squares and splines. These schemes center mainly on the notions of "splines under tension" [9]. Following a calculus of variations approach, these techniques basically find splines that minimize tension energy under different "tension parameters," similar to a stiff band under a parameterized amount of bending tension passing through knot points. A difficulty in this approach is finding a meaningful interpretation of a "tension parameter" for a decision-maker. How should a value be determined?

A second problem with splines under tension as a means of fitting a preference function is that the resulting curve may still overshoot the required $[0,1]$ boundary. Even when the slopes at the boundary are set equal to zero, the spline may still have an overshoot problem because the overshoot may help minimize the tension energy (see Figure 2-3). An interpolation scheme for fitting preference functions must eliminate such overshoots.

I present axioms about human decision making that I choose to adopt for the

Table 2.1: Elicited preference values for maximum stress.

Stress	μ
200 MPa	1.00
210 MPa	0.95
225 MPa	0.50
230 MPa	0.10
250 MPa	0.00

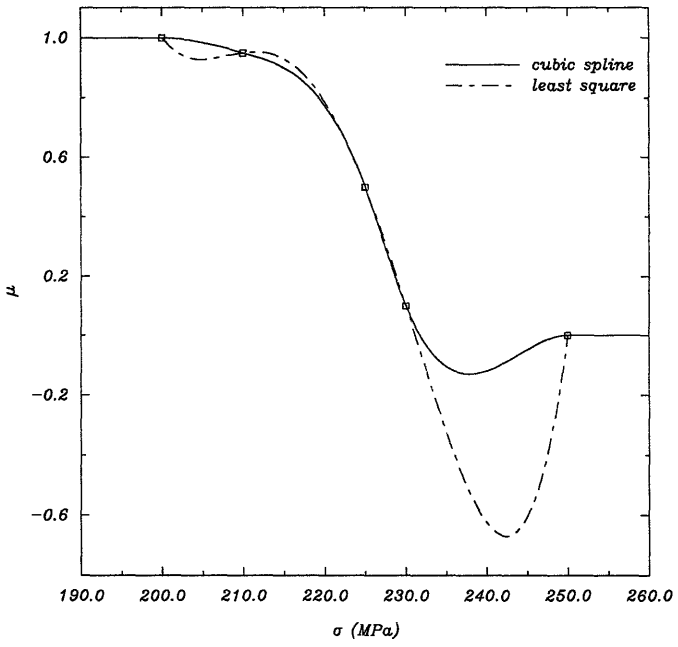

Figure 2-2: Graph of preference function using common interpolation schemes.

purposes of the developed interpolation scheme. These may be excessively restrictive for all types of human decision-making, which is beyond our purposes. Instead, I seek to lay out explicitly the axioms which must be adopted to use this interpolation scheme (the strong continuity assumptions, for example). I then demonstrate resulting properties of the derived preference functions.

Axiom 1 *A preference function is a numerical scale bounded in* $[0, 1]$.

Axiom 2 *Among the set of points considered by a designer, at least one will be a utopian point with preference 1, and at least one more will be an unacceptable point of preference 0.*

These axioms are actually the foundation for using measurement theory to elicit preference values. The worst and best cases are assigned preference value 0 and 1. The rest are assigned preference values between 0 and 1.

To fit a preference function to a set of elicited preference values, the level of continuity in the resulting function must be known. I choose to fit a smoothly varying curve to the preference data. This implies that a designer's choice among values is smoothly varying.

Axiom 3 *The rate that a designer changes preference for values in a domain is continuous.*

This axiom implies that a preference function is first derivative continuous and thus the function itself is differentiable. This implies that:

Proposition 1 *If* $\mu(x)$ *is 0 or 1, then* $\mu'(x) = 0$.

Proof. Either points in the right- or left-hand neighborhood of x have the same value of μ, or x is a local maximum or minimum for x. In the first case, $\mu'(x)$ must be zero since μ is differentiable and μ' is zero in the neighborhood of x. In the second case, $\mu'(x)$ is zero since x is a local maximum or minimum. ■

This means that when fitting a preference function to some data points, I will assume that the slope at the end-points of support of the preference function ramps up slowly. I will also assume the preference is smooth about any peak points.

Another important feature of approximate reasoning is in dealing with fuzzy numbers, when the decision-making involves grades of preference in sets. I choose not to restrict to fuzzy numbers. However, if a designer elicits preference data which has the fuzzy-convex property, I will insist on maintaining the fuzzy-convex property across all of the domain.

These restrictions form a precise statement of what I seek from an interpolation function. I seek to find

$$\mu : \mathbb{R} \rightarrow [0, 1]$$

such that

1. $\mu(x) \in [0, 1] \ \ \forall\, x$,
2. μ is differentiable,
3. $\mu(x_i) = \mu_i$ for a finite set of known pairs $\{(x_1, \mu_1), \ldots, (x_n, \mu_n)\}$.
4. If the set of known pairs $\{(x_1, \mu_1), \ldots, (x_n, \mu_n)\}$ is a fuzzy-convex set, then μ is fuzzy-convex.

McAllister and Roulier [26, 27] developed an algorithm which produces a monotonicity and convexity preserving second-degree Bernstein polynomial. Further, it is very fast and efficient to implement, and requires no user-judgment to adjust any "tension parameter" values as do the spline under tension techniques.

The ability of this method to preserve the boundedness and monotonicity of a preference function depends on the following property of Bernstein polynomials on an interval.

Let $\vec{d_i} = (x_i, \mu_i)$ and $\vec{o_i} = (t_i, \tilde{\mu}_i)$ be arbitrary points. Let $\vec{w_i} = (a, b)$ be an arbitrary point with $a = (x_i + t_i)/2$. Let g be the piece-wise linear spline passing through the points $\vec{d_i}$, $\vec{o_i}$ and $\vec{w_i}$ with a single discontinuity at a, as shown in Figure 2-4. Let $B_2[\vec{d_i}, \vec{w_i}, \vec{o_i}]$ be the second-degree Bernstein polynomial of g on $[x_i, t_i]$. That

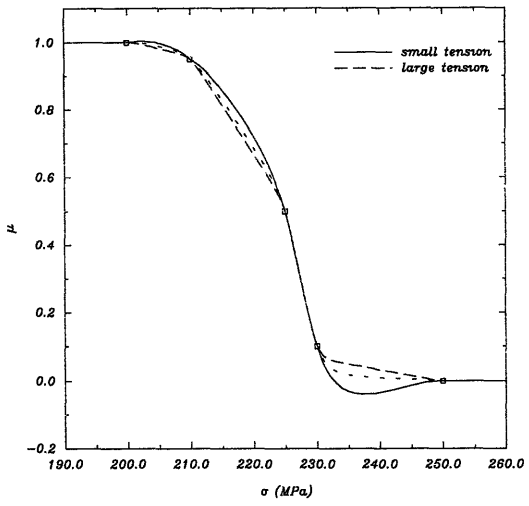

Figure 2-3: Interpolation using a spline in tension. The different curves correspond to different "tension levels."

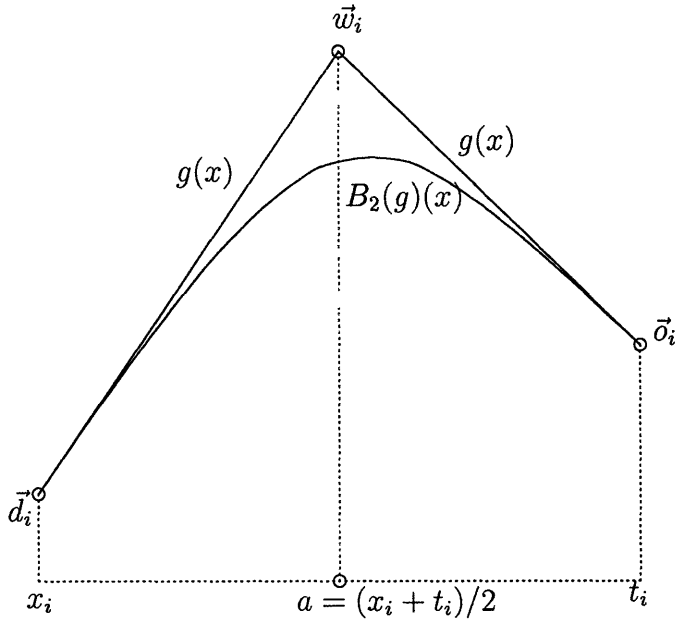

Figure 2-4: A second-order Bernstein polynomial with middle knot that has abscissa $a = (x_i + t_i)/2$.

is,

$$\begin{aligned} B_2[\vec{d_i}, \vec{w_i}, \vec{o_i}](x) &= B_2(g)(x) \\ &= \frac{g(x_i)(t_i - x)^2 + 2b(x - x_i)(t_i - x) + g(t_i)(x - t_i)^2}{(t_i - x_i)^2}. \end{aligned}$$

Proposition 2 *The following properties hold:*

1. $B_2(g)(x_i) = g(x_i) = \mu_i$, $B_2(g)(t_i) = g(t_i) = \tilde{\mu}_i$;
2. $B_2'(g)(x_i) = g'(x_i)$, $B_2'(g)(t_i) = g'(t_i)$;
3. *if g is monotone on $[x_i, t_i]$, then $B_2(g)$ is monotone on $[x_i, t_i]$;*
4. *if g is convex (concave) on $[x_i, t_i]$, then $B_2(g)$ is convex (concave) on $[x_i, t_i]$.*

Proof. Proofs for the above properties 1 and 2 are trivial. Properties 3 and 4 are shown in [27], but are essentially derived from the "convex hull" property of any quadratic functions. Hence $B_2(g)$ "preserves the shape" of g, and $B_2(g)$ has a continuous first derivative on $[x_i, t_i]$. ■

This proposition forms the basis for why the algorithm presented below preserves the desired preference function properties. Given proper data points, a quadratic Bernstein polynomial will preserve the local behaviour. A quadratic curves needs 3 coefficients to specify the curve. If one attempts to fit a quadratic curve directly to the points $(x_i, \mu_i), (x_{i+1}, \mu_{i+1})$, however, the prescription of the slope at x_i would prescribe the slope at x_{i+1}, which in turn may force the interpolation not to behave as a preference function.

The solution proposed is to augment the given set of points $\{(x_1, \mu_1), \ldots, (x_n, \mu_n)\}$ with additional points $(t_i, \tilde{\mu}_i)$ which are positioned to ensure the satisfaction of the preference function properties. The problem then becomes to find proper positioning of these additional points.

The complete proposed algorithm to fit a preference function to a set of preference data $\{(x_1, \mu_1), \ldots, (x_n, \mu_n)\}$ is as follows:

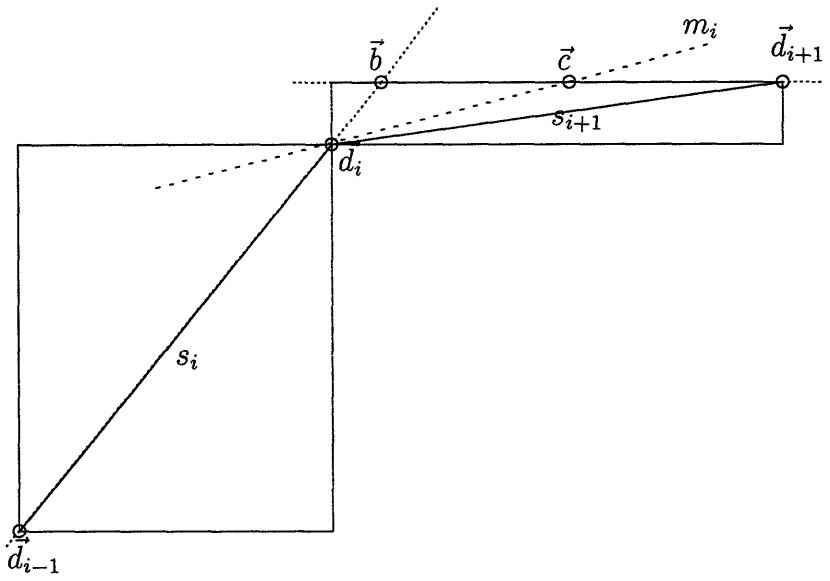

Figure 2-5: Construction of slope m_i complying with the local convexity and monotonicity.

1. First determine the slope m_i at each known point (x_i, μ_i). To do this, first define

$$s_i = (\mu_i - \mu_{i-1})/(x_i - x_{i-1}).$$

The following properties of m_i must be met:

(a) m_i must be consistent with the monotonicity and convexity of the piecewise linear function determined by the data points (x_{i-1}, μ_{i-1}), (x_i, μ_i), (x_{i+1}, μ_{i+1}).

(b) m_i must vary continuously with respect to changes in s_i and s_{i+1} when the signs of s_i and s_{i+1} agree (*i.e.*, when (x_i, μ_i) is not a extremum with fixed slope of zero).

(c) Only one knot should be required between two data points. This is to minimize the complexity of the algorithm.

The construction of m_i given below satisfies the above properties.

- If $s_i s_{i+1} \leq 0$, then set $m_i = 0$. This guarantees that local extremes of the

data points are assigned slopes 0. This also segments the entire data into monotonically non-increasing or non-decreasing subsets.

- If $|s_i| > |s_{i+1}| > 0$, then extend a line through $\vec{d_i} = (x_i, \mu_i)$ with slope s_i until it intersects the horizontal line through $\vec{d}_{i+1} = (x_{i+1}, \mu_{i+1})$ at the point $\vec{b} = (b_x, \mu_{i+1})$. Refer to Figure 2-5. Then define

$$c_x = \frac{b_x + x_{i+1}}{2}, \tag{2.1}$$

which is the abscissa of point $\vec{c}$ shown in Figure 2-5. Slope m_i at (x_i, μ_i) is defined as

$$m_i = \frac{\mu_{i+1} - \mu_i}{c_x - x_i}. \tag{2.2}$$

Note that

$$c_x > \frac{x_i + x_{i+1}}{2}. \tag{2.3}$$

- If on the other hand, $0 < |s_{i+1}| < |s_{i+1}|$, we then reverse the above procedure by extending the line through (x_i, μ_i) with slope m_{i+1} until it intersects the horizontal line through (x_{i-1}, μ_{i-1}) at the point (b_x, μ_{i-1}). Then we set $c_x = (b_x + x_{i-1})/2$ and $m_i = (\mu_i - \mu_{i-1})/(x_i - c_x)$. The end point slopes m_0 and m_n are set to zero explicitly since this is required by Proposition 1.

2. We now insert a knot point between each x_i and x_{i+1} and fit a Bernstein polynomial to the $2n - 1$ data points.

   Let R_i be the rectangle determined by the points (x_i, μ_i) and (x_{i+1}, μ_{i+1}) and let the midpoint segment of it be the line segment that bisects R_i vertically and is bounded within each R_i. Refer to Figure 2-6. Let L_i be the line that passes through (x_i, μ_i) with slope m_i.

   Without loss of generality we assume nondecreasing data points; for non-increasing data sets, the same algorithm can be applied without change; we use non-decreasing data points for demonstration purposes. There are two distinct cases

regarding the intersection of the neighboring slope lines L_i and L_{i+1}, depending on whether the knots change the local convexity of the spline or not.

- Case 1. L_i and L_{i+1} intersect at a point $\vec{z} = (t_i, z_y)$ in R_i. Refer to Figure 2-6. Let

$$\vec{v_i} = (x_i + \frac{x_i + t_i}{2},\ \mu_i + L_i\frac{x_i + t_i}{2}) \tag{2.4}$$

$$\vec{w_i} = (x_{i+1} - \frac{x_{i+1} + t_i}{2},\ \mu_{i+1} - L_{i+1}\frac{x_{i+1} + t_i}{2}) \tag{2.5}$$

  as shown in Figure 2-7. Let L be the line joining $\vec{v_i}$ and $\vec{w_i}$ and define

$$\tilde{\mu}_i = L(t_i). \tag{2.6}$$

  Now we will let $\vec{o_i} = (t_i, \tilde{\mu}_i)$ be a knot for the spline. Refer to Figure 2-7. Define μ on $[x_i, x_{i+1}]$ as follows:

$$\mu(x) = \begin{cases} B_2[\vec{d_i}, \vec{v_i}, \vec{o_i}](x) & \text{on } [x_i, t_i], \\ B_2[\vec{o_i}, \vec{w_i}, \vec{d}_{i+1}](x) & \text{on } [t_i, x_{i+1}]. \end{cases} \tag{2.7}$$

  From Proposition 2 we immediately see that $\mu(x) \in C^1[x_i, x_{i+1}]$ and satisfies

$$\begin{aligned} \mu(x_i) &= \mu_i, \\ \mu(x_{i+1}) &= \mu_{i+1}, \\ \mu'(x_i) &= m_i, \\ \mu'(x_{i+1}) &= m_{i+1}. \end{aligned}$$

  Moreover, if the first-degree spline defined by the points $\vec{d_i}$, $\vec{v_i}$, $\vec{w_i}$, and $\vec{d}_{i+1}$ is convex (concave) and/or monotone, then μ is convex (concave) and/or monotone.

- Case 2. L_i and L_{i+1} do not intersect within R_i. As a consequence of Proposition 3 (shown below), there are only two possible situations as depicted in Figure 2-8. The knot $\vec{o_i}$ is determined similarly as in Case 1, but with $t_i = (x_i + x_{i+1})/2$. The definitions of $\vec{v_i}$, $\vec{w_i}$, $\tilde{\mu}_i$, $\vec{o_i}$, and μ remain

as in (2.4), (2.5), (2.6), and (2.7), respectively. Refer to Figure 2-9. Then $\mu \in C^1[x_i, x_{i+1}]$ and μ preserves the shape of the data. Note here that the knot $\vec{o}_i$ becomes the point at which the convexity changes.

This algorithm discusses non-decreasing data points for demonstration with the figures; however, it remains equally valid for non-increasing data sets. Also, by slicing the whole data set at the local maximums and minimums, the whole data set consists of many independent segments, where each is non-increasing or non-decreasing.

We now prove the proposition that the slope assigned at $\vec{d}_i$ complies with the local convexity and monotonicity.

Proposition 3 *Consider data points which are non-decreasing. The line L_i of slope m_i passing through $\vec{d}_i = (x_i, \mu_i)$ intersects the midpoint segments of both adjoining rectangles R_{i-1} and R_{i+1}.*

Proof. For $m_i = 0$, the assertion is trivial since the line of slope 0 through (x_i, μ_i) always forms a horizontal edge to each of the adjacent rectangles and thus intersects the midpoint segment of each at the end point. Therefore we only need to consider the cases where $m_i \neq 0$. Since the slope assignment in the above step is symmetrical, we only need to consider the case of an interior point (x_i, μ_i) with

$$m_i > 0$$

and

$$0 < s_{i+1} < s_i,$$

as depicted in Figure 2-5. From (2.1), (2.2), (2.3), We see that

$$x_{i+1} > c_x > \frac{x_i + x_{i+1}}{2}. \tag{2.8}$$

With this and (2.2) we have

$$s_{i+1} < m_i < 2s_{i+1} \tag{2.9}$$

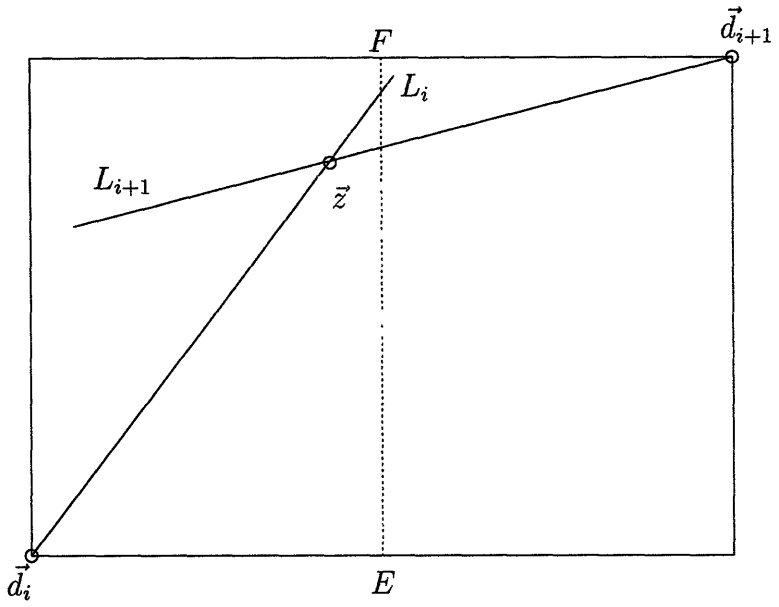

Figure 2-6: L_i and L_{i+1} intersect within R_i.

and

$$0 < m_i < s_i. \tag{2.10}$$

Inequality (2.9) means that the line L_i through (x_i, μ_i) of slope m_i intersects the midpoint segment of R_i. The second inequality (2.10) means that L_i also intersects the midpoint segment of R_{i-1}. Refer to Figure 2-10. ■

A feature of the inserted knot points $\{(t_i, \tilde{\mu}_i)\}$ which will be required subsequently is their monotonicity with their neighbors.

Proposition 4 *The points $(t_i, \tilde{\mu}_i)$ determined in the algorithm are monotonic with (x_i, μ_i) and (x_{i+1}, μ_{i+1}).*

Proof. Without loss of generality, we consider a set of monotonically increasing data. According to Proposition 3 the slopes assigned must all be non-negative.

In case 1, this proof is trivial since the intersection $\vec{z}$ of L_i and L_{i+1} is inside R_i. Thus $\vec{v}_i \in \overline{d_i z} \in R_i$ and $\vec{w}_i \in \overline{z d_{i+1}} \in R_i$. The knot $\vec{o}_i$ thus satisfies

$$\vec{o}_i \in \overline{v_i w_i} \in R_i.$$

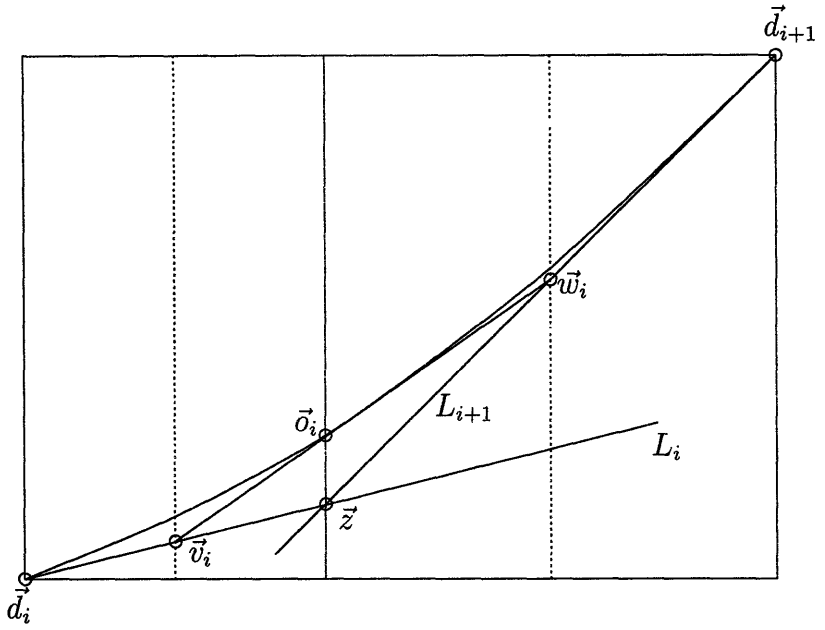

Figure 2-7: Determination of $\vec{o}_i$ for Case 1.

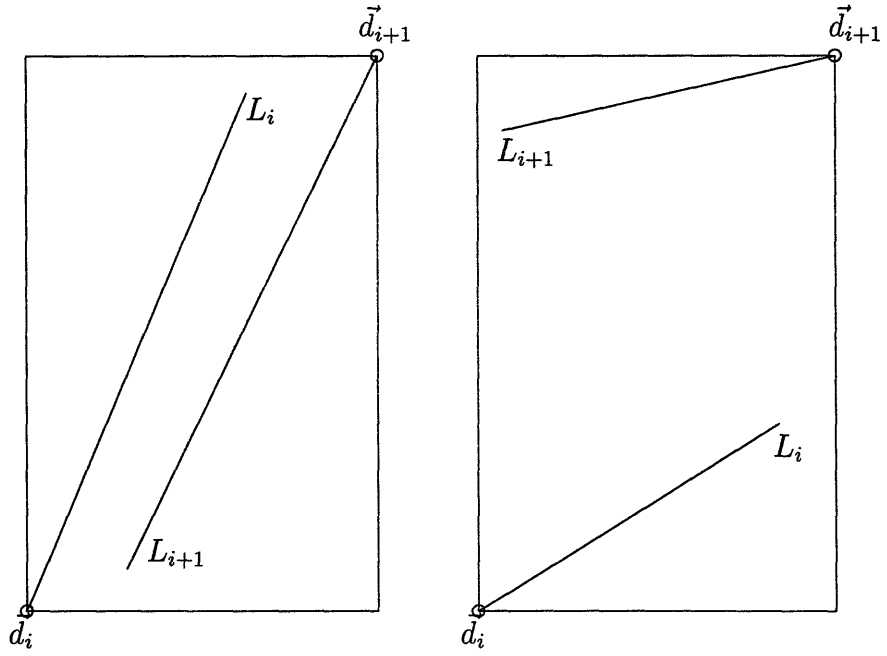

Figure 2-8: Two possible cases when L_i and L_{i+1} do not intersect within R_i.

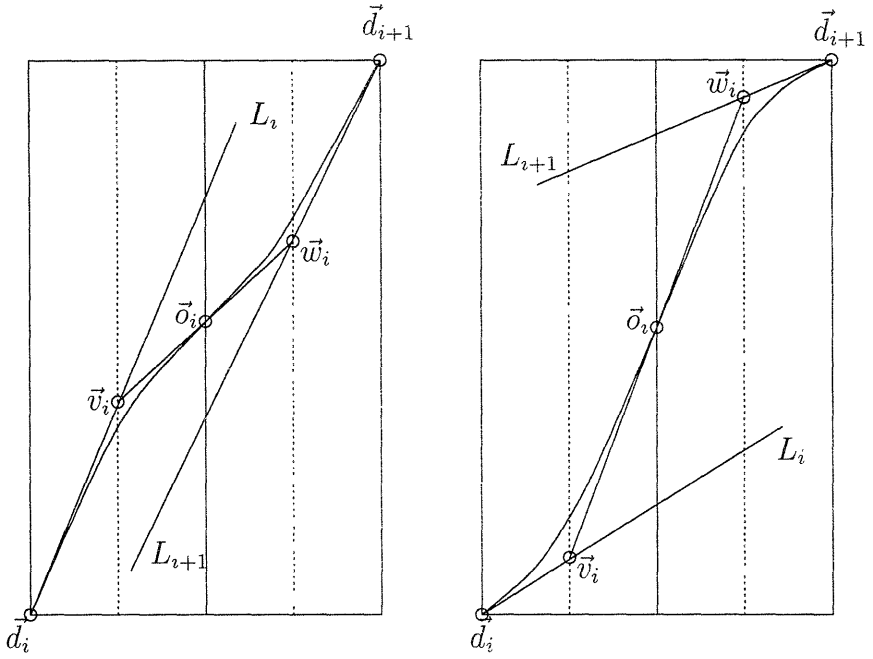

Figure 2-9: Determination of $\vec{o}_i$ for Case 2.

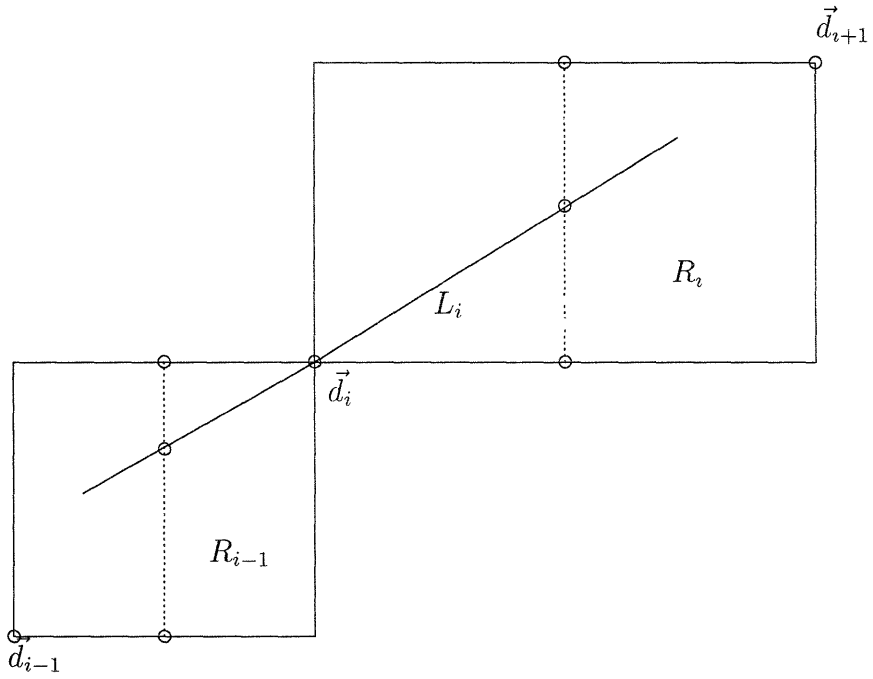

Figure 2-10: Line L_i intersects the midpoint segments of R_i and R_{i+1}.

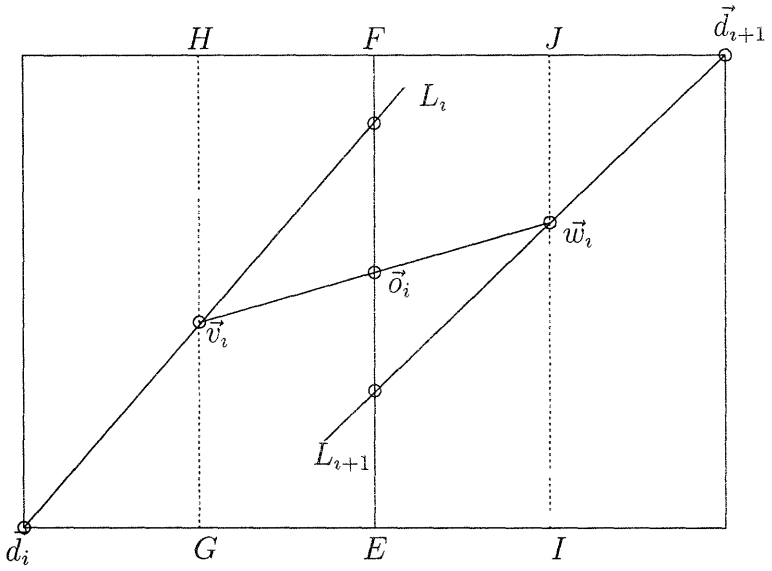

Figure 2-11: L_i must intersect with $\overline{GH}$ and L_{i+1} must intersect with $\overline{IJ}$.

For case 2, as shown in Figure 2-11, L_i has to intersect midpoint segment $\overline{EF}$ and hence it also has to intersect the line segment $\overline{GH}$. We have $\vec{v}_i \in R_i$. By the same argument L_{i+1} has to intersect line $\overline{IJ}$. Thus

$$\vec{v}_i \in \overline{GH} \in R_i,$$

$$\vec{w}_i \in \overline{IJ} \in R_i,$$

and so

$$\vec{o}_i \in \overline{v_i w_i} \in R_i.$$

■

2.3.4 Fuzzy-Convex Preference Functions

Of particular interest is the situation when the elicited data satisfies the fuzzy-convex property.

Proposition 5 *Given a fuzzy-convex set of elicited-preference data, the interpolated preference function determined using the algorithm of Section 2.3.3 will be fuzzy-*

convex.

Proof. By assumption, the elicited preference data is monotonically increasing on $\{x_0, \ldots, x_k\}$ and monotonically decreasing on $\{x_k, \ldots, x_n\}$. Using the construction of the algorithm in Section 2.3.3, we insert a point $\vec{o_i} = (t_i, \tilde{\mu}_i)$ monotonic with $(x_i, \mu_i), (x_{i+1}, \mu_{i+1})$, by Proposition 4. Then using the algorithm in Section 2.3.3, a monotonic g (as in Proposition 2) is defined on $[x_i, t_i]$ and on $[t_i, x_{i+1}]$, consisting of L_i and L_{i+1} respectively. Given this, then by Definition (2.7) and Proposition 2, μ is monotonic between $[x_i, t_i]$ and $[t_i, x_{i+1}]$, which with the monotonic t_i implies μ is monotonic on $[x_i, x_{i+1}]$. But together with the monotonic data points, this implies μ is monotonically increasing on $[x_1, x_k]$ and monotonically decreasing on $[x_k, x_n]$. ■

Thus, the interpolation scheme is very suitable for imprecise calculations and decision-making in a fuzzy environment.

2.3.5 Convergence Properties of the Algorithm

Another concern which arises is the behaviour of the interpolation when additional elicited points $\{x_i, \mu_i\}$ are added for interpolation. The preference function should become more accurate as any arbitrary point $\{x_i, \mu_i\}$ is added. This is in contrast with some simple interpolation schemes, which can become worse, for example, if the points then become unevenly spaced. The algorithm of Section 2.3.3 exhibits no such properties, and will converge to the actual preference function.

Denote by $\mu_n(x)$ an interpolated preference function derived using the algorithm of Section 2.3.3, with n elicited data points. Denote by $\mu_a(x)$ the theoretical elicited-preference function if the interval scale question were asked on all points $x \in \mathbb{R}$.

Proposition 6

$$\lim_{n \to \infty} \mu_n(x) = \mu_a(x)$$

Proof. Let $B[a, b]$ be the space of bounded continuous functions on $[a, b]$ and J be the interval $[x_0, x_{n-1}]$ and J_i be the interval $[x_{i-1}, x_i]$. By Axiom 1 and 3, $\mu_a(x) \in$

$B[x_0, x_{n-1}]$. By Proposition 4, all the points $\{t_i, \tilde{\mu}_i\}$ are contained inside the rectangles $\{R_i\}$ and thus $\mu_a(x) \in B[x_0, x_{n-1}]$. The metric between μ_n and μ_a is

$$\begin{aligned} d(\mu_n, \mu_a) &= \max |\mu_n(x) - \mu_a(x)| \\ &= \max_i (\max_{x \in J_i} |\mu_n^i(x) - \mu_a(x)|) \\ &< \max_i H_i \\ &= H^*. \end{aligned}$$

Here H_i is the height of R_i and H^* is the largest of H_i. When $n \to \infty$, intervals $J_i \to 0$ and

$$\lim_{n \to \infty} H^* = 0$$

since $\mu_a(x)$ is continuous. Thus

$$\lim_{n \to \infty} d(\mu_n, \mu_a) = 0$$

and

$$\lim_{n \to \infty} \mu_n(x) = \mu_a(x).$$

■

Thus, one can start with a few sample points of elicited preference, do engineering calculations, and interpolate to fit the results. One can then add points as necessary to increase the accuracy of the calculations. But most importantly, the points can be added at any additional $x \in \mathbb{R}$, and the accuracy will not decrease. The points need not be evenly spaced, for example. This is important for engineering calculations, since this allows one to focus on critical regions by eliciting preference from points there, with the confidence that the interpolated preference will not become skewed. Again, the interpolation scheme is very suitable for imprecise calculations and decision-making in a fuzzy environment.

As an example, consider a fuzzy number whose preference function is given exactly

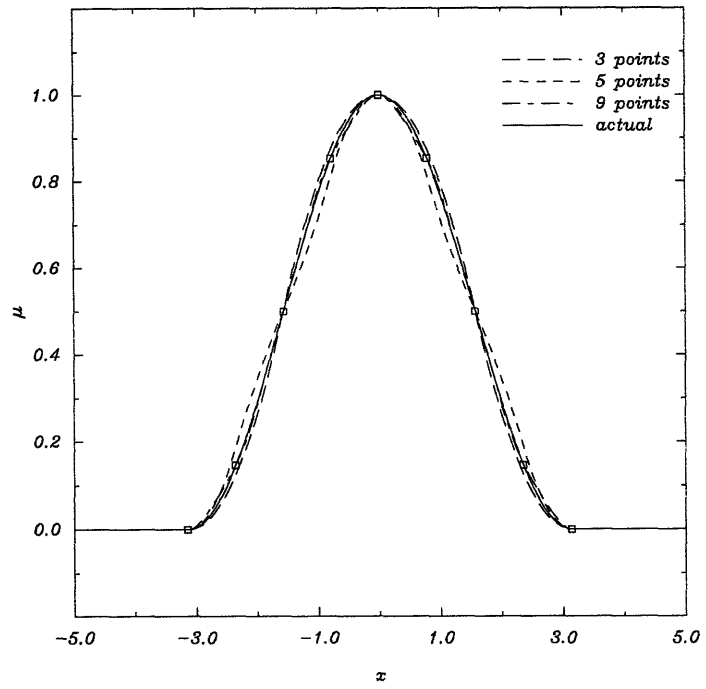

Figure 2-12: Convergence to a differentiable preference function.

by

$$\mu_a(x) = \begin{cases} \frac{1}{2}(\cos(x)+1) & x \in [-\pi, \pi] \\ 0 & \text{otherwise} \end{cases}$$

but this expression remains unknown to the decision maker. The interval scale construction can be used to find a finite number of points to approximate μ_a. This is shown in Figure 2-12 for different numbers of points.

As a non-differentiable example, consider a preference function given exactly by

$$\mu_a(x) = \begin{cases} 1 & x < 0 \\ -\frac{1}{3}x^3 + \frac{3}{2}x^2 - \frac{11}{6}x + 1 & x \in [0, 3] \\ 0 & x > 3 \end{cases}$$

but again this expression remains unknown to the decision maker. The interval scale construction can be used to find a finite number of points to approximate μ_a. This is shown in Figure 2-13 for different numbers of points.

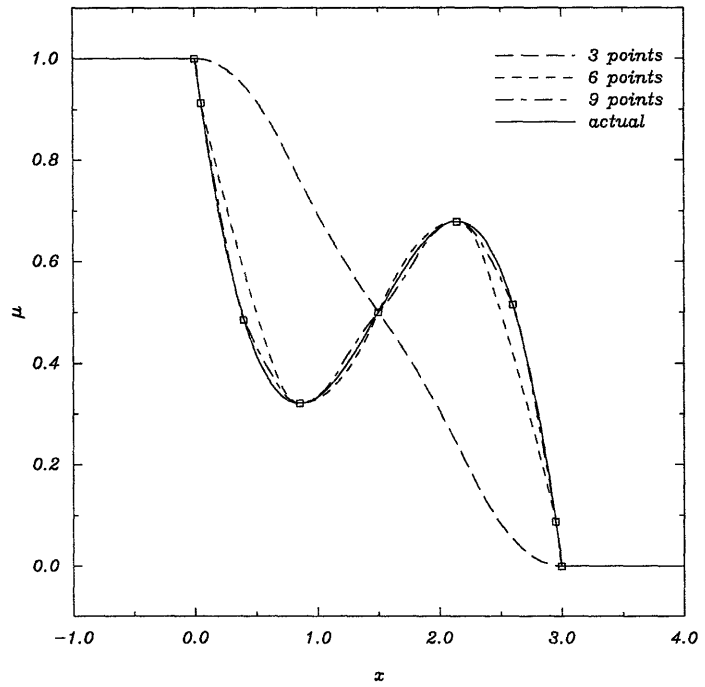

Figure 2-13: Convergence to a non-differentiable preference function.

Finally, notice that the method satisfies our requirement for a simple and efficient implementation. Quadratic interpolation is minimal to satisfy the differentiability requirement across the entire curve. Thus, the method satisfies all of the stated needs of fuzzy mathematics, but also does so in an efficient manner.

Chapter 3

Propagating Preference through Systems of Constraints

3.1 Introduction

After a formal model has been constructed, a designer has only imprecise specifications on the variables in the model. He/she may want to partially specify the model and observe the restrictions on other unspecified variables. A designer may specify particular values on *some* of the variables, simply to observe the effects of the partial specification.

For example, a designer may wish to conjecture values for design variables, and observe the restrictions on performance variables. In the design of a car front-wheel suspension, selecting particular ranges of values for the design geometry will result in a range of possible caster angles. Performing this kind of calculation will allow a designer to interpret the performances achievable and sensitivity to each design variable. The designer explores the design space, especially the portion in which the final design is most likely to emerge. This could be viewed as a *sufficiency* test and exploration.

On the other hand, a designer may wish to lock values of design performance, and observe the restrictions over the design configurations that can be used to achieve it. In the design of a car front-wheel suspension, locking the caster angle within an

acceptable small range will restrict the design geometry. Performing such calculations will allow a designer to find out what design configurations should be pursued if the performances are to be achieved. This can be viewed as a *necessity* test and exploration.

3.2 Constraint Systems

constraint-based CAD systems are used in mechanical engineering to design engineered products [1, 29, 55]. However, these systems are crisp constraint systems, in that the constraints are strictly reinforced. System response to a point in solution space is either "on" (the point satisfies the constraints) or "off" (the point does not satisfy the constraints). It is proposed to build imprecise constraint systems, in which constraints are imprecisely maintained. With increased dimensionality of solution space, a valid solution is more likely to emerge. A set approach can be employed.

3.2.1 Crisp Constraint Systems

In crisp constraint systems, engineering models are represented as variables in a system of relations:

$$\begin{aligned} f_1(x^1, \ldots, x^n) &= y^1 \\ f_2(x^1, \ldots, x^n) &= y^2 \\ &\vdots \\ f_m(x^1, \ldots, x^n) &= y^m \end{aligned} \tag{3.1}$$

Typically, x^i are known as the independent or input variables, and y^j are known as the dependent or output variables. The map $\vec{f}$ relating the two sets of variables $\vec{x}$ and $\vec{y}$ could be an equation, a computer program, an expert system, or any means of evaluating performance $\vec{y}$ given a design configuration $\vec{x}$.

Given a configuration, a designer typically performs calculations to rate different values of the configuration variables. For example, the maximum bending stress in a structure might be calculated, since the designer must ensure that the bending stress

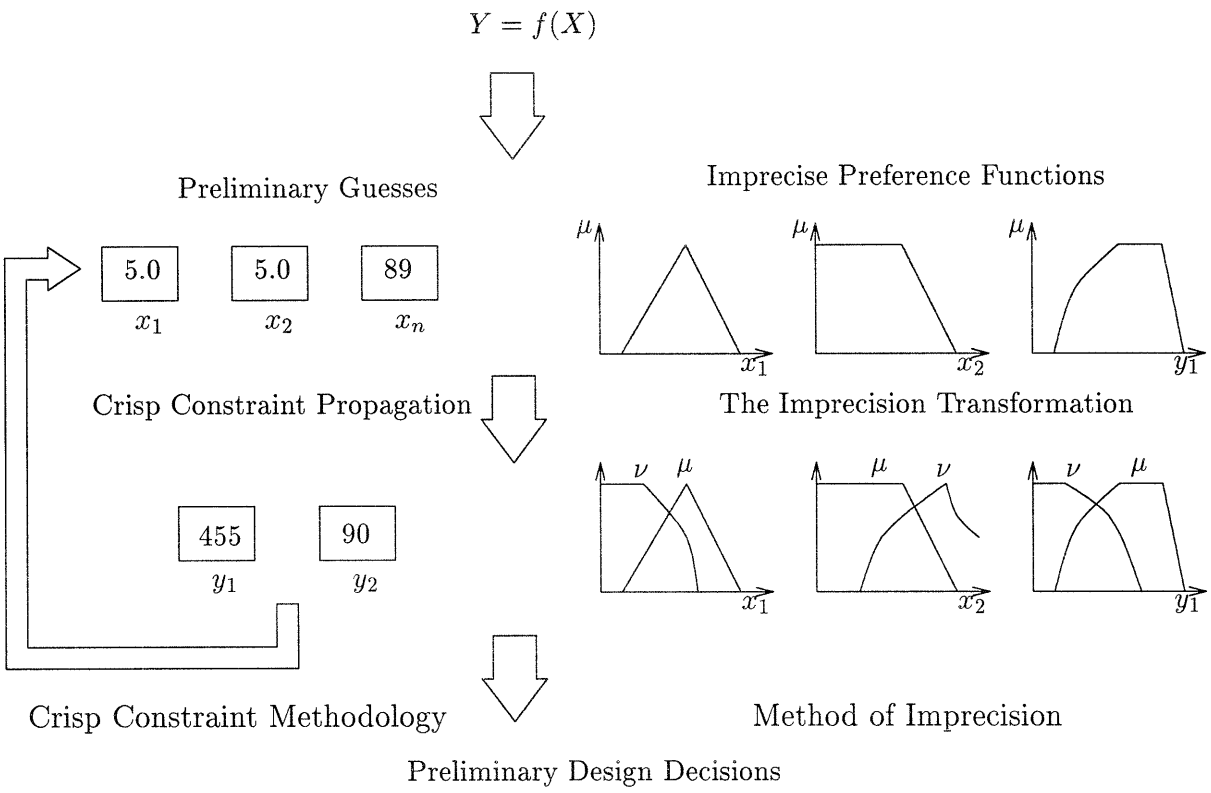

Figure 3-1: constraint-based CAD systems.

is not excessive.

Thus, in constraint-based CAD systems, the system relations are typically formulated as input/output relations as in (3.1), but the system of relations is not used as an input/output system. In many engineering applications, the input variables are not selected and the output variables values observed. Rather, many times the dependent variables must be fixed, and the input variables adjusted to the fixed output variable values. On the other hand, the exact values of the output variables are usually not known, but rather imprecise boundaries are. Typically, users iterate between desired input variable selections and allowed output variable values.

This paradigm is depicted in Figure 3-1[1]. A user may select a subset of the variables in $x^1, \ldots, x^n$, $y^1, \ldots, y^m$ to fix, and propagate these specifications onto the remaining variables, to observe their effect on the model.

Consider now a relation which has all of its values fixed by a user inconsistently, *i.e.*, all of the variable values in one relation are simultaneously specified by the user, but the relation does not hold with these values. The typical action taken by crisp constraint propagation systems in this condition is to flag a warning to the user that a variable must be relaxed [55].

3.2.2 Imprecise Slackening of Crisp Constraints

When dealing with over-constrained systems one often wishes to relax some of the constraints. The way *imprecise slackening* works is very much like adding slack variables in linear optimization problems. Suppose there is a constraint

$$x^2 + y^2 = 4.$$

The imprecise-constraint approach is to add another variable, say,

$$z = x^2 + y^2.$$

[1]This figure is from "Propagating Imprecise Engineering Design Constraints" by K. N. Otto and E. K. Antonsson, Prepared for Proceedings of the Fourth International Conference on Fuzzy Systems, 1995, Yokohama, Japan.

Then a preference function is defined on z, with maximum preference peaking at $z = 4$ and ranging from $z = 3.8$ to $z = 4.2$. The crisp constraint is thus slackened; now it has a fuzzy boundary. The constraint value $x^2 + y^2$ can be anywhere on [3.8, 4.2], but values around 4 are preferred. Original crisp constraint can be thought of as a δ function in this context. Crisp constraints become imprecise if the δ function is relaxed. The degree of relaxation depends on the willingness of the designer to violate the crisp constraints and the trade-off among all conflicting constraints.

3.2.3 Imprecise Constraint Systems

Here it is proposed to use the system of relations as described above in conjunction with the propagation of not just single values, but entire sets of values using imprecise preference. Graphically, the proposed computational model is depicted in Figure 3-1. The user makes estimates of values for any of the input and output variables desired, represented with preference. The preference specifications are then induced onto the remaining variables, to observe the *a priori* restrictions on the remaining variables.

After having made the induced preference calculations, the user can observe the imprecise performance achievable and proceed to judge the model. Thus, use of imprecise quantities within constraint systems offers the ability to shift much of the iterative searching a user must do onto the computational platform, by computing many sets of values simultaneously.

A further benefit is gained from using imprecise quantities. When propagating single values through constraint systems, if a relation has all of its variable values simultaneously specified and the values are such that the relation does not hold, the system is over-constrained. This condition can be overcome by imprecise mathematics. The imprecisely specified variables cause the system to be imprecisely over-constrained, rather than totally over-constrained. If a relation has all variables specified, the imprecise mathematics can be used to restrict the preference functions beyond what was originally specified by the user. This is analogous to constraint systems warning users of inconsistent crisp values.

The next section will develop the requisite mathematics for these calculations.

3.3 Propagating Preference within Constraint Systems

In this section we seek to develop proper representation for constrained systems and develop algorithms for propagating constraints.

3.3.1 Modeling Constrained Systems

For engineering design problems, products can be represented with models represented as variables in a system of relations, such as in Equation 3.1, which can be rewritten as

$$\vec{y} = \vec{f}(\vec{x}). \tag{3.2}$$

To develop the mathematics for propagating through constraint systems in the most general sense, we generalize the performance map (3.2) to a constraint system relating variables among an input space X and an output space Y:

$$\vec{F}(\vec{z}) = \vec{0}. \tag{3.3}$$

The total imprecise space is thus $Z = X \times Y \in \mathbb{R}^n \times \mathbb{R}^m = \mathbb{R}^N$. $\vec{F}$ is a set of equality constraints relating the variables in Z space.

During an iterative design process, usually some variables are imprecisely specified while the remaining variables remain unspecified. This means that Z can be partitioned into a set of specified variables S and a set of unspecified variables U. Thus, (3.3) can be written as

$$\vec{F}(\vec{s}, \vec{u}) = \vec{0}, \tag{3.4}$$

where $\vec{s}$ and $\vec{u}$ are elements in sets S and U, respectively. Define the pre-image in S of each $\vec{u} \in$ U using $\vec{F}$,

$$\Gamma(\vec{u}) = \{\vec{s} \in S \mid \vec{F}(\vec{s}, \vec{u}) = \vec{0}\}. \tag{3.5}$$

Thus $\Gamma(\vec{u})$ is the set of all solution points $\vec{s}$ in S which *simultaneously* satisfy all

equations in the constraint network $F(\vec{s}, \vec{u}) = \vec{0}$ for an unspecified value $\vec{u}$ in U. The dimension of $\Gamma(\vec{u})$ could be 0, finite, or infinite, depending on the solvability of (3.5).

Given this, the imprecise constraints in the form of preference functions μ placed on S can be induced onto U. This will show the induced restrictions on the unspecified space U. To do this, we define an *induced preference* on U,

$$\nu(\vec{u}) = \sup \{\mu(\vec{s}) \mid \vec{s} \in \Gamma(\vec{u})\}. \tag{3.6}$$

This is nothing more than Zadeh's extension principle generalized from explicit mappings to implicit constraint equations. We denote it ν to distinguish it from the independent imprecise specifications μ, since both μ and ν co-exist on any variable in the constraint system.

3.3.2 Degree of Freedom for Propagation in Constraint Systems

Constraint propagation through systems of equations is not a simple exercise in search. Systems of equations can be defined that may have no solutions. To explore this, rewrite (3.4) as

$$\begin{aligned} F_1(s_1, \ldots, s_\gamma, u_1, \ldots, u_\beta) &= 0 \\ F_2(s_1, \ldots, s_\gamma, u_1, \ldots, u_\beta) &= 0 \\ &\vdots \\ F_m(s_1, \ldots, s_\gamma, u_1, \ldots, u_\beta) &= 0. \end{aligned} \tag{3.7}$$

Here γ is the dimension of S and β is the dimension of U. Preference on S can be propagated onto U, as discussed above. However, restrictions must be placed on β for the the constraint propagation to have meaning. Suppose there is a point $\vec{s}^* \in$ S, for which the combined preference is defined as

$$\mu(\vec{s}^*) = \min[\mu(\vec{s}_1^*), ..., \mu(\vec{s}_\gamma^*)]. \tag{3.8}$$

To find the restrictions of specified variables on unspecified variables, one must know all sets of $\vec{u}$ which satisfy (3.7). Thus one needs to solve the equations

$$\begin{aligned} F_1(s_1^*, \ldots, s_\gamma^*, u_1, \ldots, u_\beta) &= 0 \\ F_2(s_1^*, \ldots, s_\gamma^*, u_1, \ldots, u_\beta) &= 0 \\ &\vdots \\ F_m(s_1^*, \ldots, s_\gamma^*, u_1, \ldots, u_\beta) &= 0, \end{aligned} \tag{3.9}$$

which is a set of m equations with β unknowns. We define *degree of freedom for propagation* as $\beta - m$. This is slightly different from normal notions of system degrees of freedom, in that we consider m, the number of unspecified variables, not $n + m$, the total number of variables.

From an analogy with linear systems, the following observations are made, assuming all equations are independent.

- If $m < \beta$, the system is under-constrained for propagation. Typically, infinite means of propagating constraints exist. There are fewer constraints than variables, and so there are multiple means to propagate the constraints. For example, consider a simple constraint

$$s_1 + u_1 + u_2 = 0. \tag{3.10}$$

  The degree of freedom is 1 since $\beta = 2$ (two unknowns) and $m = 1$ (one equation). There are infinite means to propagate a value from s_1. If an imprecise specification is made on s_1 by defining a preference $\mu(s_1)$, then the induced preference on u_1 and u_2 is $\nu(u_1) = \nu(u_2) = 1 \; \forall \; u_1$ and u_2. For any u_1, a u_2 can always be found that satisfies (3.10) together with the peak preference value of s_1.

- If $m = \beta$, the system is critically-constrained for propagation. There are a finite number of means of propagation. The degree of freedom is 0. For example,

consider the following equation,

$$s_1 + u_1 = 0. \tag{3.11}$$

With a value on s_1 specified, there is only one solution of u_1 that satisfies (3.11).

- If $m > \beta$, the system is over-constrained for propagation, with a negative degree of freedom. Typically there is no means to propagate constraints. For example, consider the following equations,

$$\begin{aligned} s_1 + u_1 &= 0 \\ s_1 - u_1 &= 0. \end{aligned} \tag{3.12}$$

For any value of s_1 (except 0) there is no solution of u_1. The induced preference on u_1 is 0 everywhere (except at 0).

3.3.3 Propagation Preference in Critically Constrained Systems

It is desired to develop a system for interacting with a system of critically constrained imprecise constraints. That is, a user should specify the design problem sufficiently to allow an algorithm to proceed, but also should not specify constraints which are inherently not solvable. For example, in a constrained system of N variables and 1 equality constraint, we seek to propagate the preference specified on $N - 1$ of the variables onto the remaining variable. Such a calculation could be repeated across all of the variables, as shown in Figure 3-2, thereby allowing a design engineer to interactively observe the specifications and the effects of other specifications at an early stage.

Previous work has found fast algorithms for single, explicit-form constraints [67]. We have developed a fast algorithm for propagating preference on arbitrary variables within a set of monotonic constraints. This will be discussed in more detail in the next section.

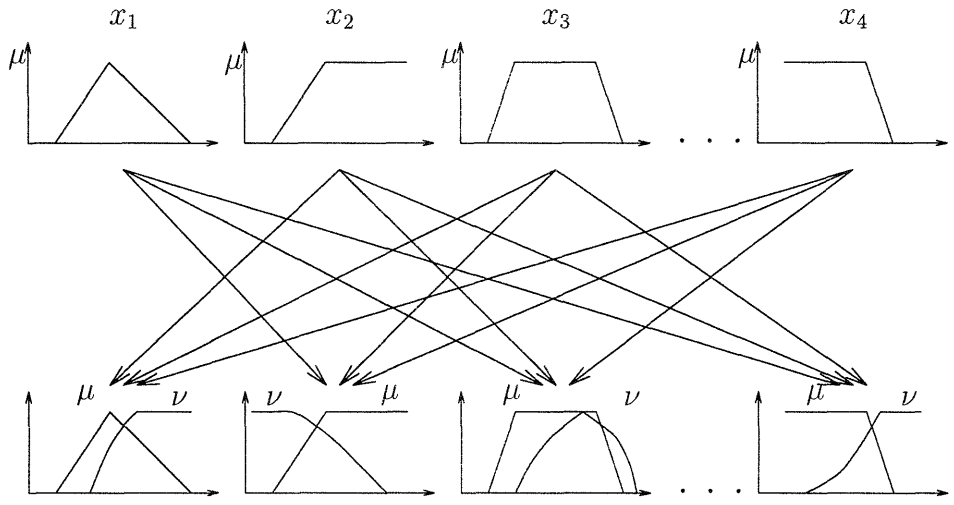

Figure 3-2: Propagation of induced preference in an n variable, 1 constraint system.

Chapter 4

ICPT Framework

4.1 Introduction

Chapter 3 has discussed the theory of propagating constraints. In this chapter the computer implementation in ICPT is presented. As an imprecise constraint tool, ICPT allows a user to manage over-constrained systems, without committing to a crisp value of a design variable in the early design stage; instead, it allows the user to specify imprecisely the values in the model.

For general cases, the propagation is done through constrained optimization. To facilitate real-time or near real-time propagation, an extension of the *Level Interval Algorithm* [67] has been developed and implemented. Input variables and output variables are treated anonymously by invoking nonlinear equation solvers; thus, propagation is made on any variables within the constraint system. This will be discussed in Sections 3 and 4. The features of the CAD tool are highlighted in Section 2.

The Imprecise Constraint Propagation Tool (ICPT) is an interactive computer tool based on the methodology in Chapter 3. It consists of a graphical user interface and an optimization core. Run in the Matlab® environment [13, 14], it invokes the Matlab Handle Graphics® [15] for its graphical applications and the Matlab Optimization Toolbox® [17] for its optimization and propagation. Matlab also supports MEX-files, which can link to outside routines [16]. As shown in Figure 4-1, the graphical interface consists of *mycmdwin.m*, *sproblem.m*, *detail.m*, *preference.m*, *con-*

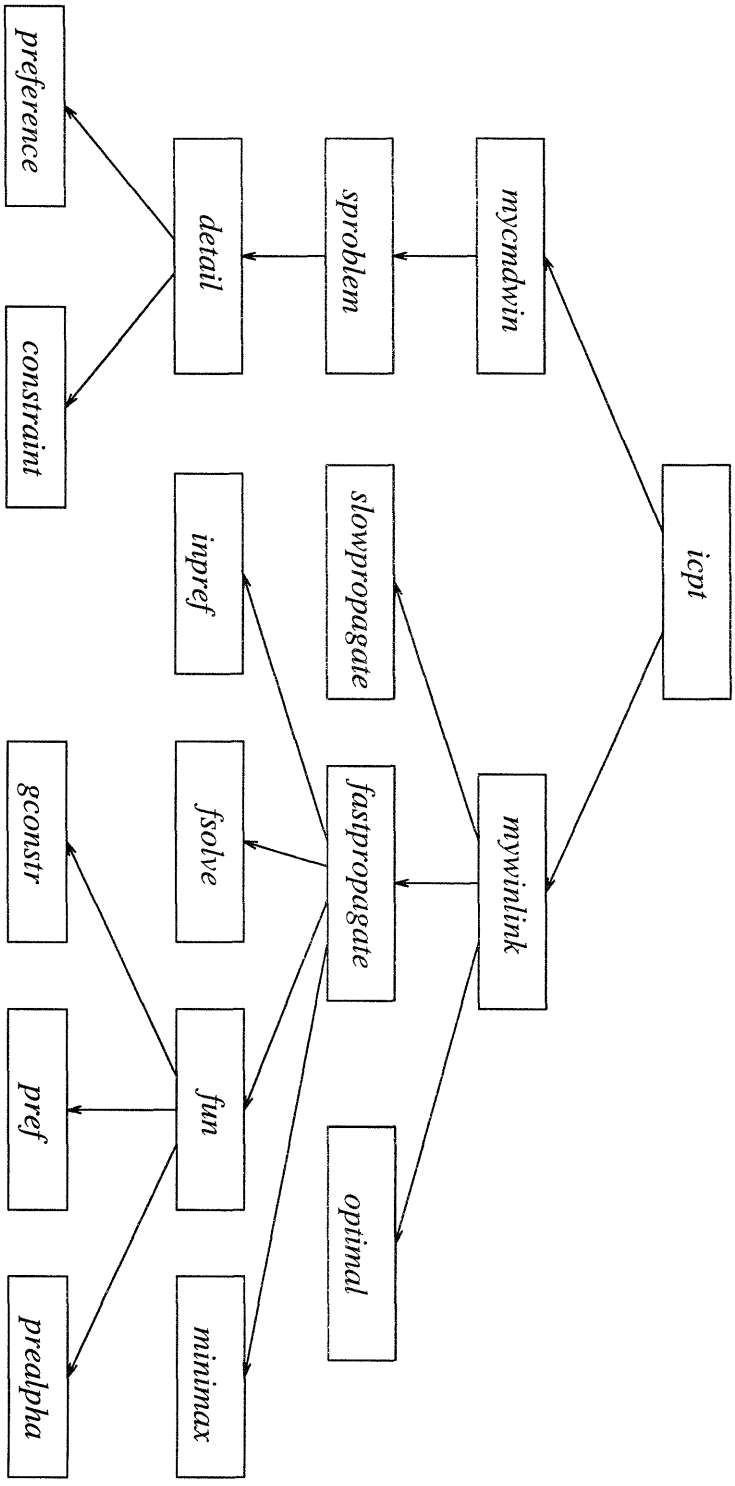

Figure 4-1: ICPT architecture.

straint.m. Each of these is a Matlab function that launches different user-interface windows. The user progressively defines the problem and inputs information that defines the problem. The optimization core consists of *slowpropagate.m* and *fastpropagate.m.* Functions *inpref.m*, *fun.m*, *gconstr.m* and *prealpha.m* manage the constraint and α-cut. Two Matlab intrinsic functions *fsolve* and *minimax* are also invoked. The main negotiation window *Concurrent Engineering Window* is launched by *mywinlink.m.* The final optimal points are determined by invoking *optimal.m.*

4.2 Graphical Interface

The graphical user-interface of the system consists of a hierarchy of pop-up text windows for defining the problem: i.e., inputting the constraints and elicited preference points. After the problem is defined, the user works within a graphics window to fine-tune the preference functions, either by dragging the preference values or by selecting different interpolation schemes (linear or quadratic) to construct the preference functions [7]. The propagation of ν onto a variable is made by clicking the corresponding button in that variable's graphics window. If the constraint on the variable is not satisfied (there is no overlapping point that has μ and ν of 1), the user can change the preference on other variables and back-propagate the newly specified preference onto the variable of concern. Thus, the influence of changing the preference on one variable can be seen by propagating onto the other variables. Through this negotiation process, less stringent preference functions are relaxed and shifted to accommodate the more stringent preferences. After the negotiation is done, the user can click the "Done" button to locate the optimal solution set, its numerical values and overall preferences will be shown in the two text windows of each axis. For a detailed description of graphical interface, see Appendix A.

4.3 Constrained Optimization

The induced preference of any *feasible* $\vec{u}^*$ in U is calculated through an optimization problem formulated as follows:

$$\begin{aligned} &\text{minimize} \quad \mu(\vec{s}) \\ &\text{subject to} \quad F(\vec{s}, \vec{u}^*) = \vec{0} \end{aligned} \tag{4.1}$$

Here the minimization is in S. The evaluation of $\mu(s)$ on S is typically defined as in (3.8). A function in the Matlab Optimization Toolbox® called *minimax*, which invokes a sequential quadratic programming (SQP) solution, can be used to optimize the above function, or more robust algorithms can be developed in its replacement.

$$\begin{aligned} &\text{minimize} \quad \max(\vec{F}(\vec{X})) \\ &\text{subject to} \quad \vec{G}(\vec{X}) \leq \vec{0} \end{aligned} \tag{4.2}$$

Here $\vec{G}(\vec{X})$ can be equality or inequality constraints.

4.4 Extension of Level Interval Algorithm (ELIA)

For systems with monotonic equality constraints, an extension of Level Interval Algorithm (LIA) [66], is implemented to facilitate rapid propagations. ELIA computes with multiple constraints rather than with only a single constraint as in LIA. It also propagates preference functions to both dependent and independent variables. Thus it solves a wider range of realistic problems, since engineering systems generally involve multiple constraints and multiple optimization goals. Instead of evaluating the dependent value directly as in LIA, ELIA solves for nonlinear equations defined by the constraints with the specified variables $\vec{s}$ fixed at a set of values $\vec{s}^*$. Given a constrained system in the form of (3.3),

$$\vec{F}(\vec{z}) = \vec{0}, \tag{4.3}$$

let the preference function on *any* γ number of these variables be specified, and β variables remain unspecified (thus $\gamma+\beta = N$). Then the overall space Z is partitioned into S and U, as described in the previous section. The constrained system becomes

$$\begin{aligned} F_1(s_1, \ldots, s_\gamma, u_1, \ldots, u_\beta) &= 0 \\ F_2(s_1, \ldots, s_\gamma, u_1, \ldots, u_\beta) &= 0 \\ &\vdots \\ F_m(s_1, \ldots, s_\gamma, u_1, \ldots, u_\beta) &= 0. \end{aligned} \tag{4.4}$$

The following steps lead to the induced preference on the remaining β variables, $\nu_1, ..., \nu_\beta$.

1. For each s_i, $i = 1, ..., \gamma$, discretize the preference function $\mu(s_i)$ into a number of α-cut values α_n, $n = 1, ..., M$, where M is the number of steps in the discretization.

2. Determine the intervals for each variable $\vec{s}_i$ at each α-cut $\alpha_n, n = 1, ..., M$.

3. Using one end-point from each of the γ intervals for each α_n, combine the end-points into an γ-ary array such that 2^γ distinct permutations exist for the array.

4. For each of the 2^γ permutations, $\vec{s}^{*,k}$, $k = 1, ..., 2^\gamma$, solve for $\vec{u}$ in the following equation.

$$\begin{aligned} F_1(s_1^{*,k}, \ldots, s_\gamma^{*,k}, u_1, \ldots, u_\beta) &= 0 \\ F_2(s_1^{*,k}, \ldots, s_\gamma^{*,k}, u_1, \ldots, u_\beta) &= 0 \\ &\vdots \\ F_m(s_1^{*,k}, \ldots, s_\gamma^{*,k}, u_1, \ldots, u_\beta) &= 0 \end{aligned} \tag{4.5}$$

   For monotonic constraints, there are a finite number of solutions. Denote the number of solutions L^k and each solution $\vec{u}^{*,k,l}$, $l = 1, ..., L^k$.

5. For each component u_j, $j = 1, ..., \beta$, the resultant interval for the α-cut, α_n, $n = 1, ..., M$, is then given by

$$u_j^{\alpha_n} = [\min_{\substack{k=1,...,2^\gamma \\ l=1,...,L^k}} (u_j^{*,k,l}), \max_{\substack{k=1,...,2^\gamma \\ l=1,...,L^k}} (u_j^{*,k,l})].$$

This formulation enables propagation to any number of β variables. The preferences on these β variables are *simultaneously* determined, reflecting that they are constrained by multiple equations.

If $m = 1$, (4.5) is reduced to

$$f(x) = F(z_1, ..., z_{n-1}, x, z_{n+1}, ..., z_N) = 0. \tag{4.6}$$

Induced preference can be computed on z_n once the preference of the other $N - 1$ variables are known. This is shown in Figure 3-2.

4.5 Discussion

A new methodology of constraint-based CAD using imprecise quantities was demonstrated through the development of an imprecise constraint propagation tool. After the specified preference functions are defined, the relations can be catalogued into three possible classes: under-constrained, critically-constrained, or over-constrained for propagation. This determination is based upon the number of independent relations and the number of imprecisely unspecified variables.

With a set of equations critically constrained for propagation, an algorithm is given which can accommodate multiple implicit equations. This is an extension from previous algorithms in the literature, which accommodate single explicit relations. Further, the algorithm can propagate preference onto any arbitrary variables within a constraint, rather than only to dependent ones.

Chapter 5

ICPT Examples

5.1 Introduction

This chapter presents two examples from mechanical engineering design. The constraints-negotiation process is highlighted and demonstrated in window scenarios taken from design sessions. Two possible techniques, namely sensitivity analysis and dimensional analysis, are proposed to speed up the computations and to simplify the problem by reducing independent design variables.

5.2 Truss Design

As an illustration of ICPT, consider the design of a frame truss, as shown in Figure 5-1. The truss is intended to support a weight W at a distance l from a wall. The width and thickness of the beam are w and t respectively. W is the applied load, and E is the elastic modulus of the material. Given a configuration, a designer typically performs calculations to rate different values of the configuration variables. For example, the maximum bending stress in the horizontal bar might be calculated, since the designer must ensure that the bending stress is not excessive. A performance relation relating the configuration values to the applied stress is given by:

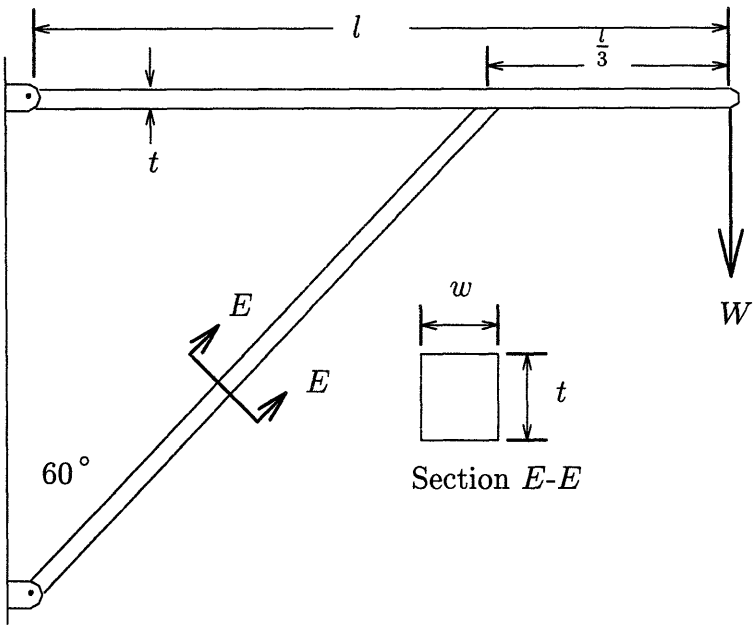

Figure 5-1: Structural Truss.

Table 5.1: Specified design variable preferences.

Variable	Low end	Optimal	High end	Unites
w	0.015	0.03	0.05	m
t	0.04	0.06	0.07	m
σ	0.25		.5	GPa
W	1.5	2.0	2.5	$10kN$
l	2	4	5	m

$$\begin{aligned} \sigma \ : \ & \mathbb{R}^4 \rightarrow \mathbb{R} \\ & (w,t,W,l) \mapsto \frac{2l(W+\frac{\rho g w t l}{6})}{wt^2}. \end{aligned} \tag{5.1}$$

The preferences on these variables are shown in Table 5.1. There are two constants, shown in Table 5.2.

A designer may select initial preference values for W, E, ρ, w, l and t. Then, one variable at a time, the induced preference from the remaining variables can be propagated onto the variable in question through 4.6. In Figure 5-2, the specified preference curves μ are in solid lines. The user may then want to see the induced

Table 5.2: Structure Truss: Constant Values.

variable	Value
g	$9.8\ \frac{m}{s^2}$
ρ	$7830\ \frac{kg}{m^3}$

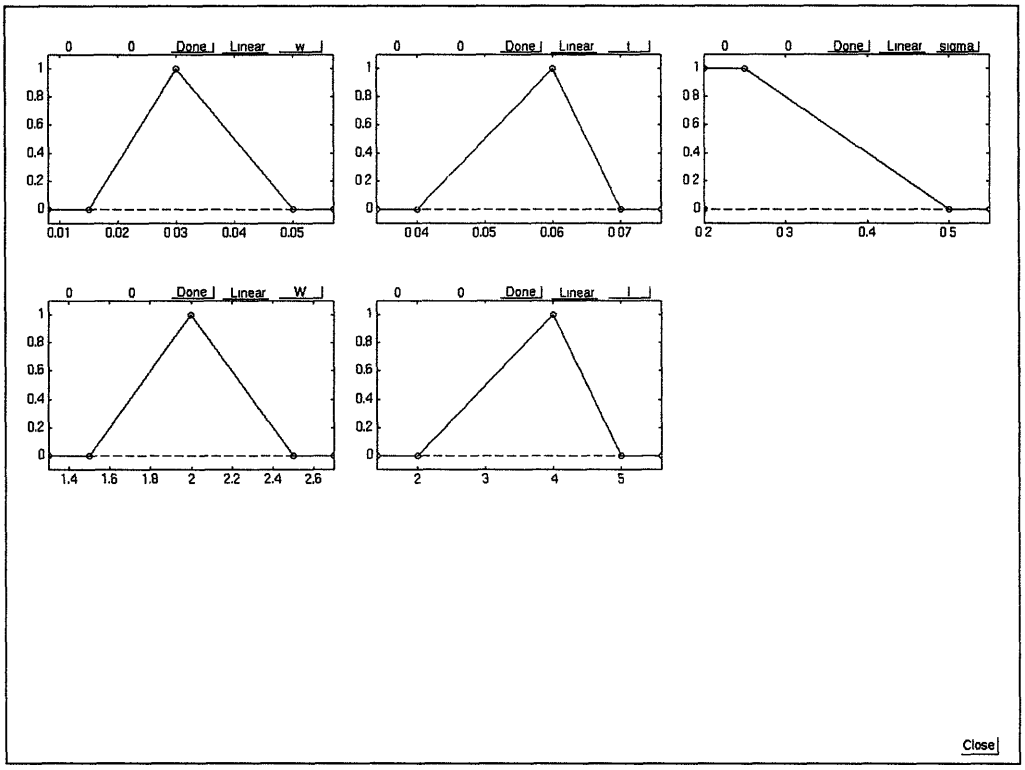

Figure 5-2: ICPT initial variable specifications. This is the graphical representation of Table 5.1. Note that crisp constraint Equation 5.1 is transferred into an imprecise constraint by allowing σ to be taken a range of values below 0.5 GPa, with indifferent full satisfaction for values below 0.25 GPa. The user input the preferences information through a user interface window (see Appendix) and can modify the curves in this graphics window by dragging the control points around.

preference ν on t, given the preference functions μ on σ, w, W and l. This is shown in Figure 5-3. The designer may realize that among all the variables, the preference ranges of W and l are actually very narrow, given the design intents that the truss is designed to hold a certain fixed weight and the length is pretty much fixed. Thus he/she then narrows the ranges on these variables around the most likely operational values, as shown in Figure 5-4.

It may be discovered that the specified preferences μ on t and w don't overlap with induced preferences ν. This is shown in Figure 5-5. To reach a satisfactory design, the designer must relax some less stringent design intents, i.e., the preferences on w and t. The designer may decide to freeze the values of W and l so that he/she can concentrate on the negotiation among w, t and τ.

A new negotiation window is then opened, shown in Figure 5-6. The designer may drag the preference curve μ toward the induced preference curve ν on t directly, or move the preference curves μ on σ and w, and observe the resulting change. The negotiation scenarios are explained in the captions of Figure 5-6 through Figure 5-16.

At any point in this process we can provide the user with feedback as to the crisp optimal solution point(s) $\vec{z}_*$. That is, the user will naturally want to find points $\vec{z}_* \in$ Z which maximize the combined preferences. In Figure 5-16, these points would be $w_* = 0.106$ (m), $t_* = 0.0727$ (m), and $\sigma_* = 0.291$ (GPa). If the user is satisfied with these values, the iterative design process stops, and the user can proceed to fabrication using these values. At any point in the iterative process of refining the imprecision, the optimal solution can be calculated and an accept/reject determination made over these points of maximum preference.

In fact, the points $\vec{z}_* \in$ Z returned are exactly related to an imprecise non-linear programming problem. If all of the variables have imprecise preference functions, then a search across X for the points $\vec{x}_*$ which maximize the overall combined preference (including those on Y) is the same as the points determined in the above, iterative approach. This was shown in [44].

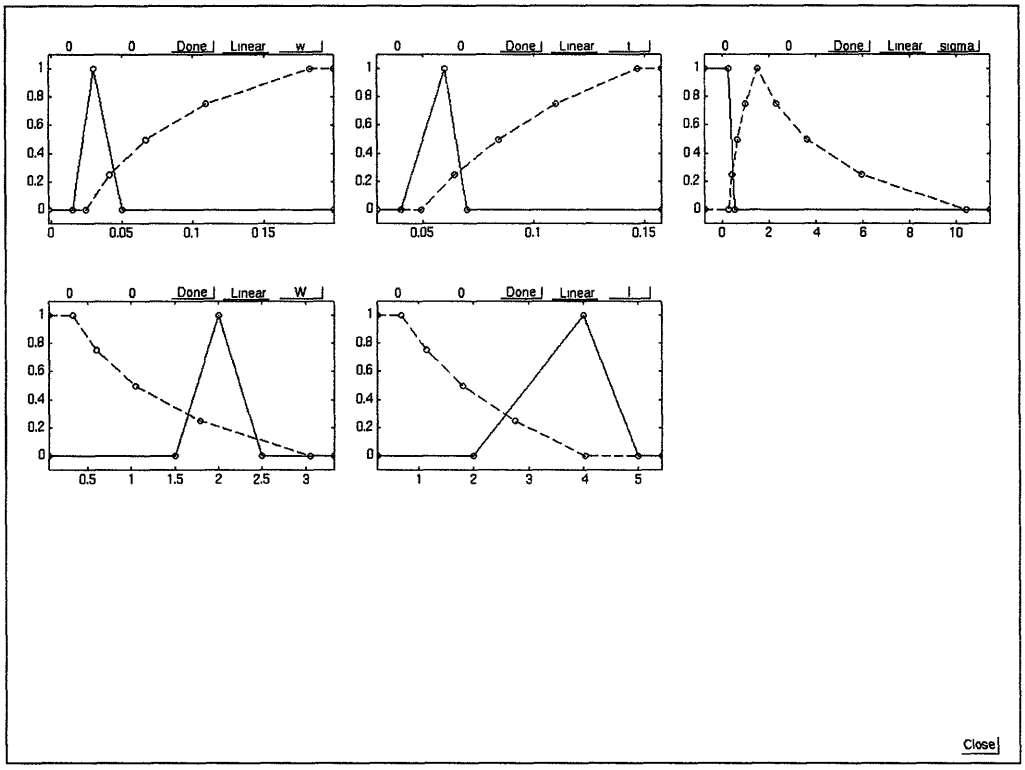

Figure 5-3: ICPT initial propagation. Above the right side of each graphics axis, there is a button on which there is a symbol representing the design variable. Pressing the button will propagate the preference of all other variables onto the variable being pressed. The dashed lines are induced preferences. Here, the user begins with very wide ranges on each variable.

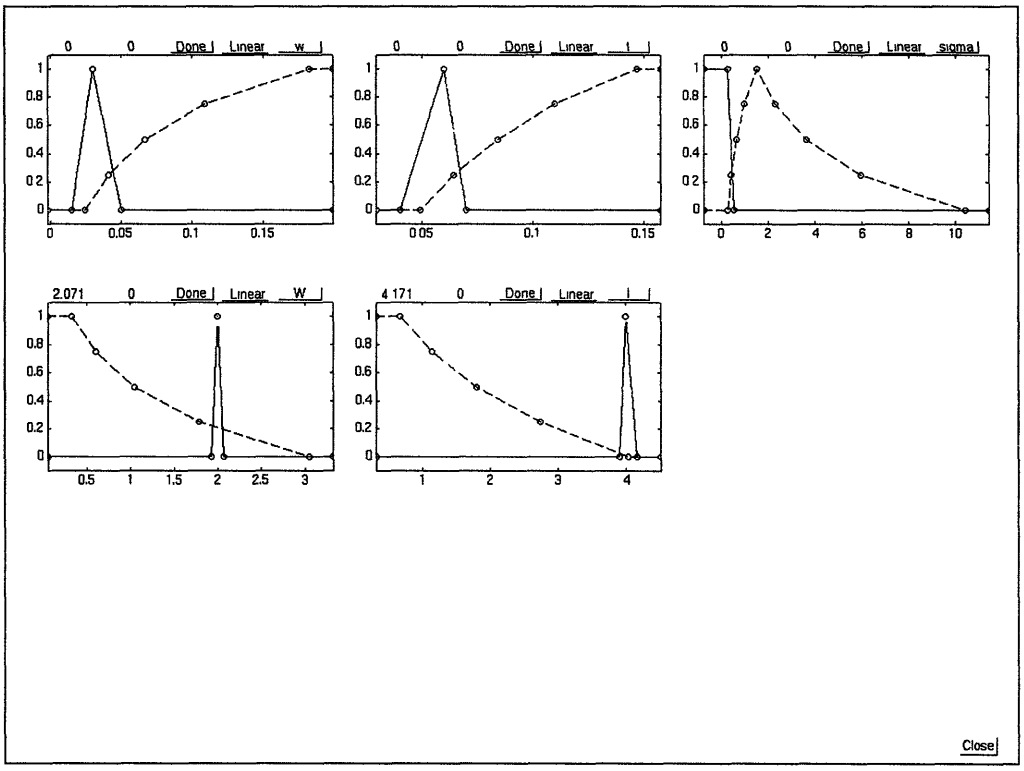

Figure 5-4: Narrowing down design variable ranges based on design intents. The designer intentionally sets wide ranges for the initial configurations to explore design space. The next step is to narrow down ranges on some of the design variables, according to design intents. Suppose the truss is to hold a certain fixed weight at a certain span, the preferences on W and l should then be narrowed, as shown here.

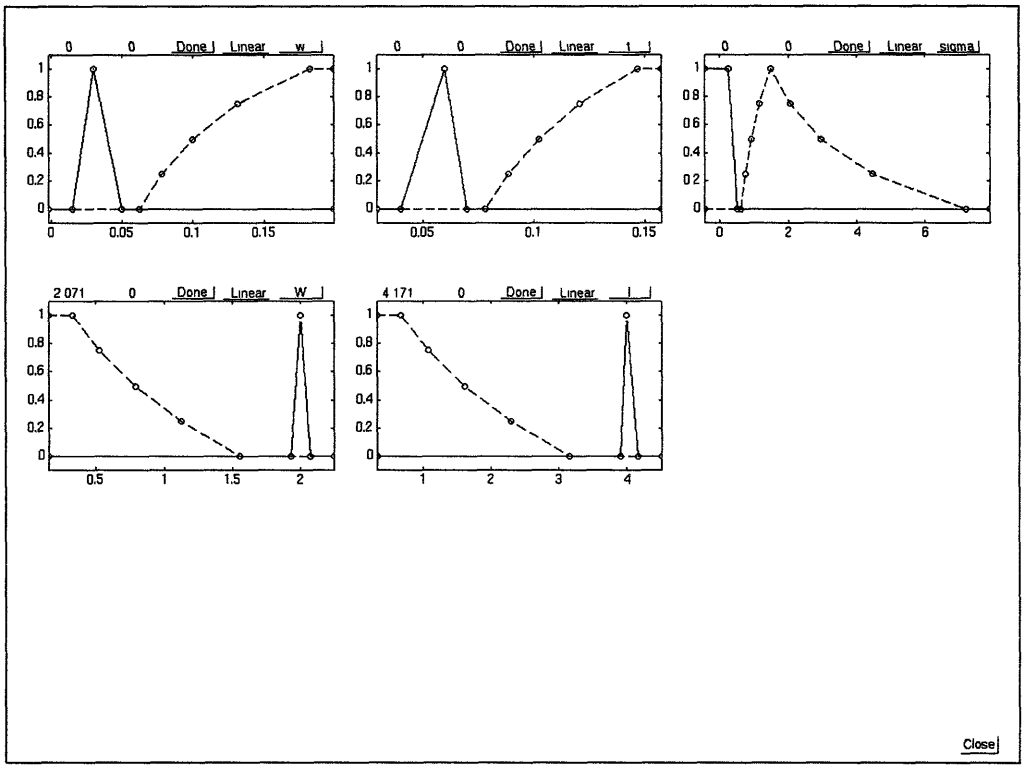

Figure 5-5: Propagation of the modified preferences. The resulting propagation yields no feasible solution: There is no overlapping interval between preference and induced preference on each design variable. Design ranges on w, t and σ have to be modified.

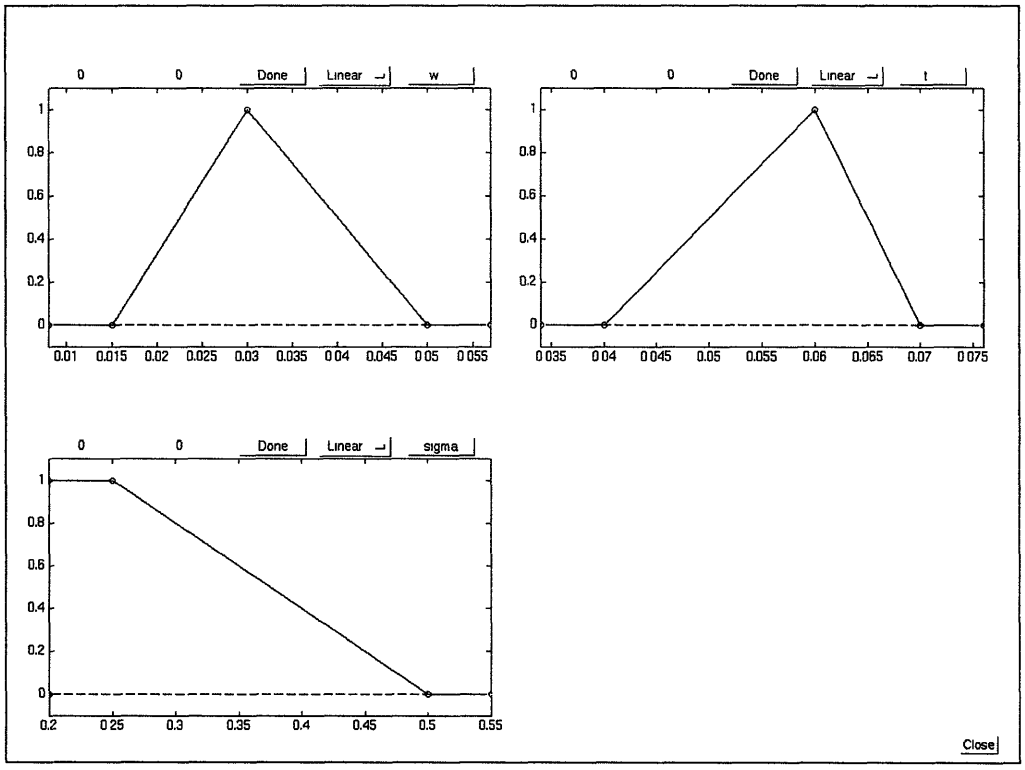

Figure 5-6: Scenario 1: New negotiation window with new set of design variables. Since W and l are almost fixed, the designer opens new design windows for the variables less constrained.

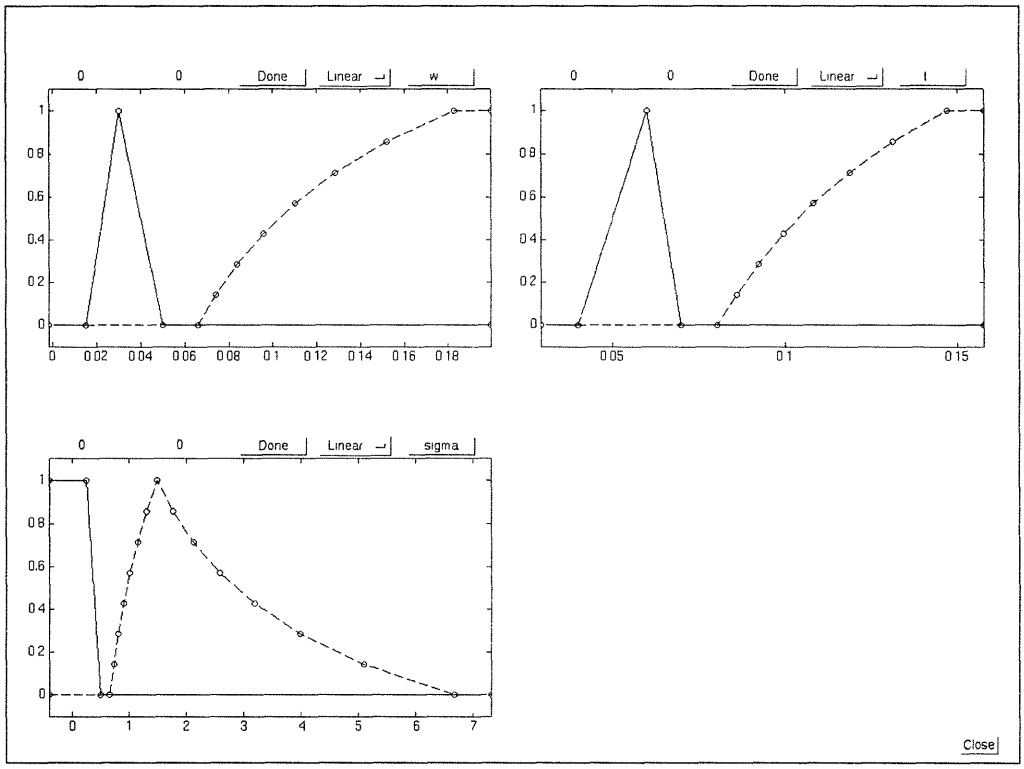

Figure 5-7: Scenario 2: Propagation of the preferences.

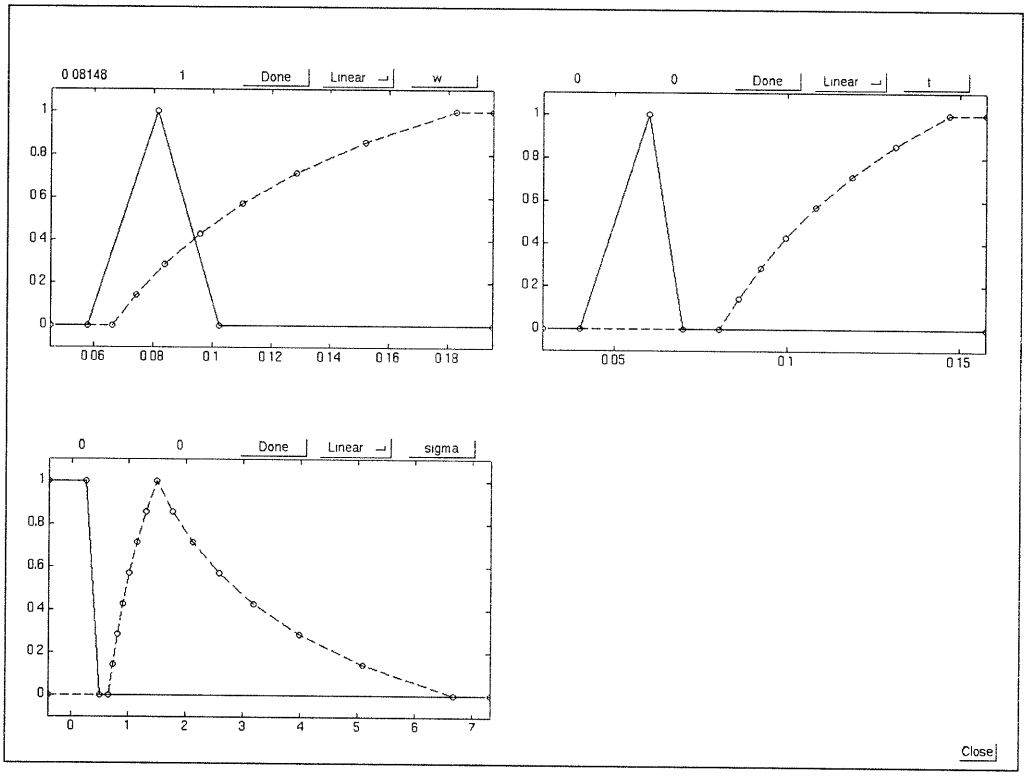

Figure 5-8: Scenario 3: Shifting preference curve on w toward induced preference curve. Realizing that using larger values of w is an acceptable design option, the designer then shifts the preference curve toward induced preference.

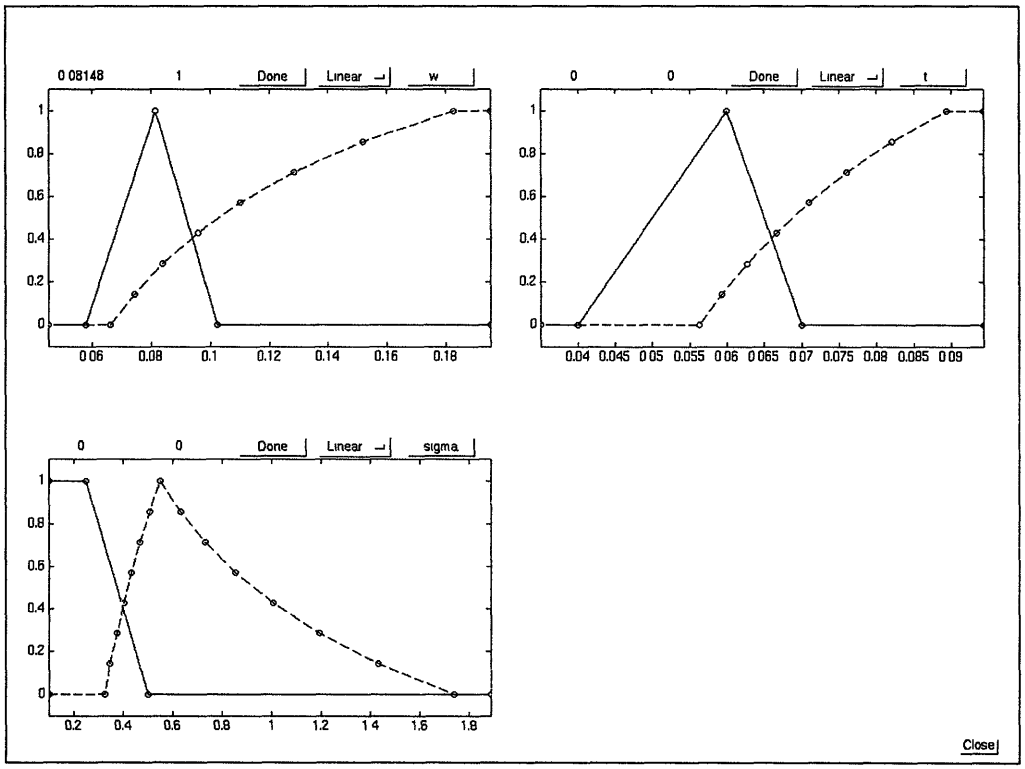

Figure 5-9: Scenario 4: The induced preferences on t and σ.

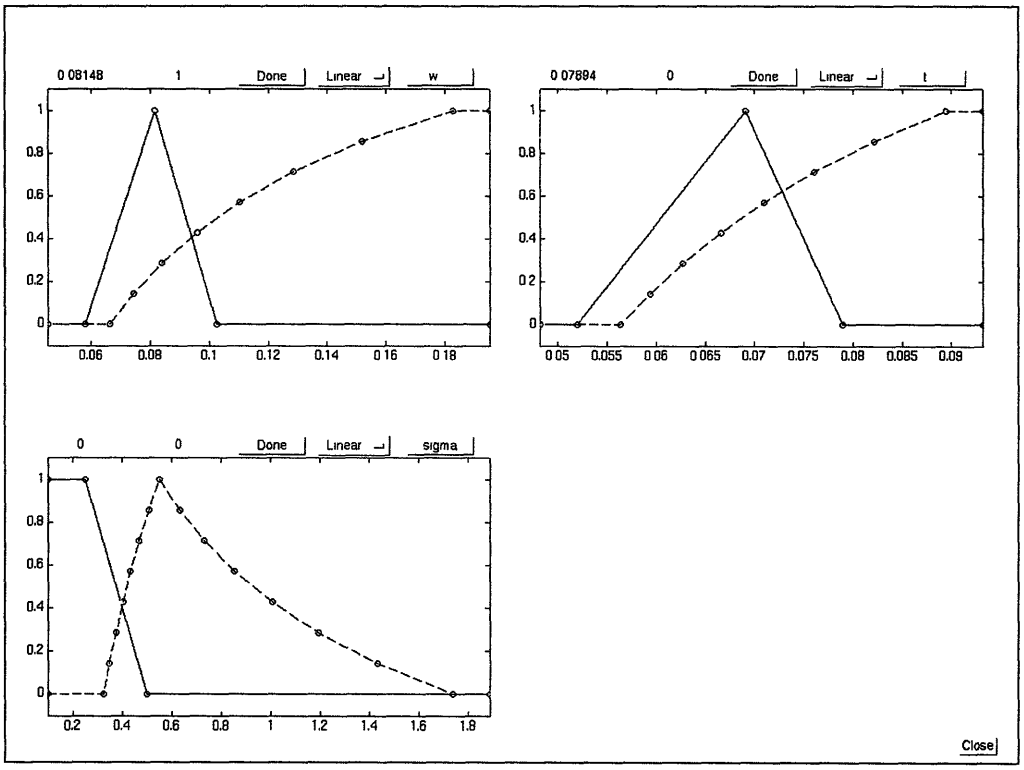

Figure 5-10: Scenario 5: The designer drags the preference curve on t further toward the region of high induced preference.

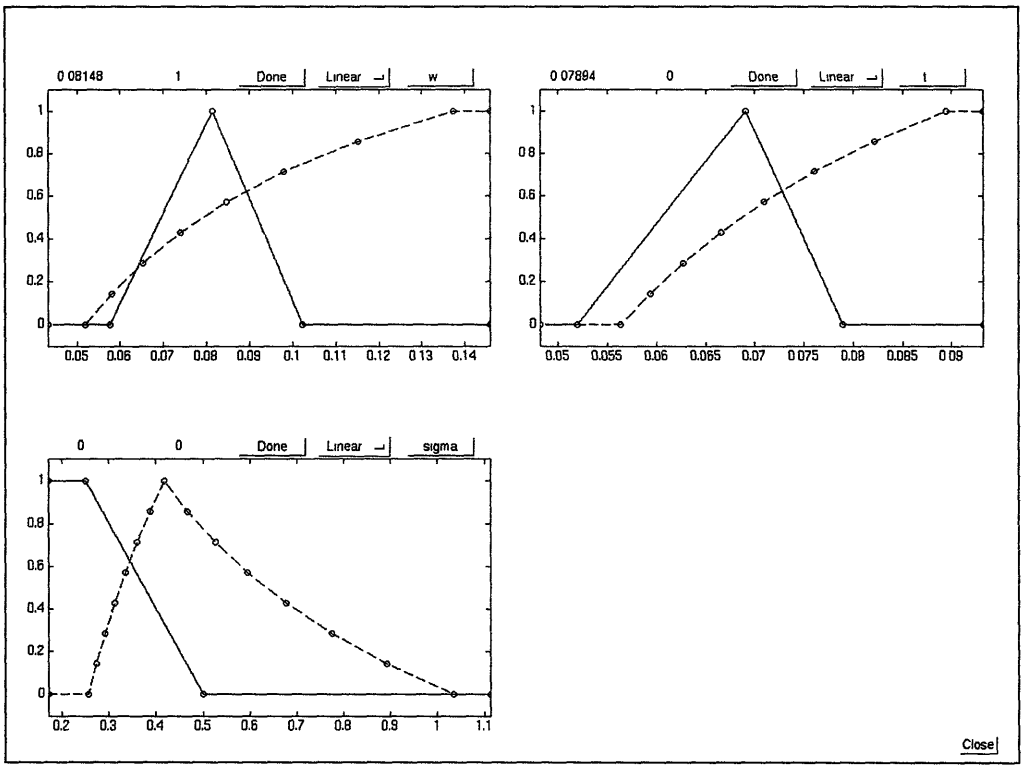

Figure 5-11: Scenario 6: The resulting propagation of the changed preference.

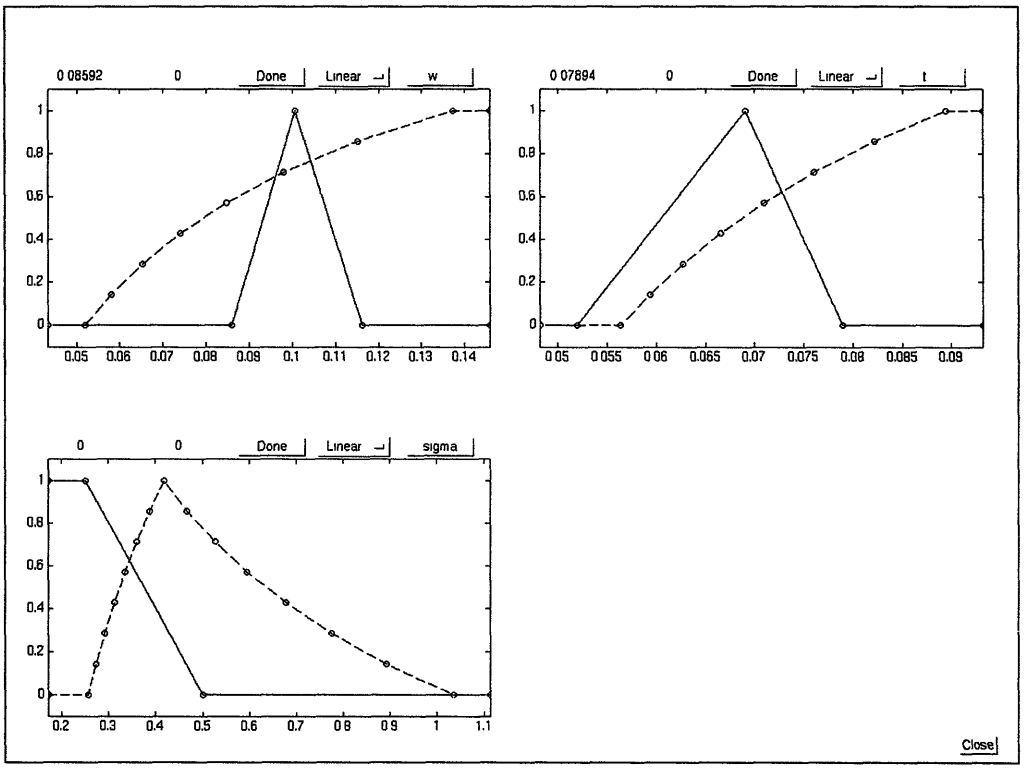

Figure 5-12: Scenario 7: The designer then moves the preference curve on w toward the high induced preference range.

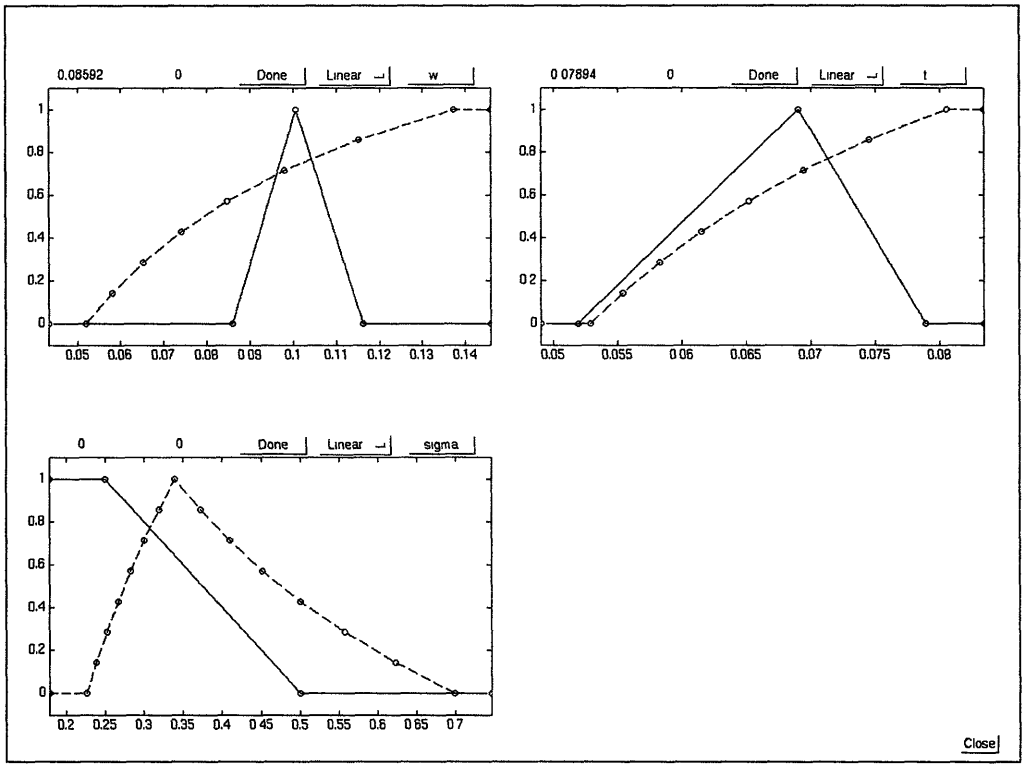

Figure 5-13: Scenario 8: This change is propagated to σ and t .

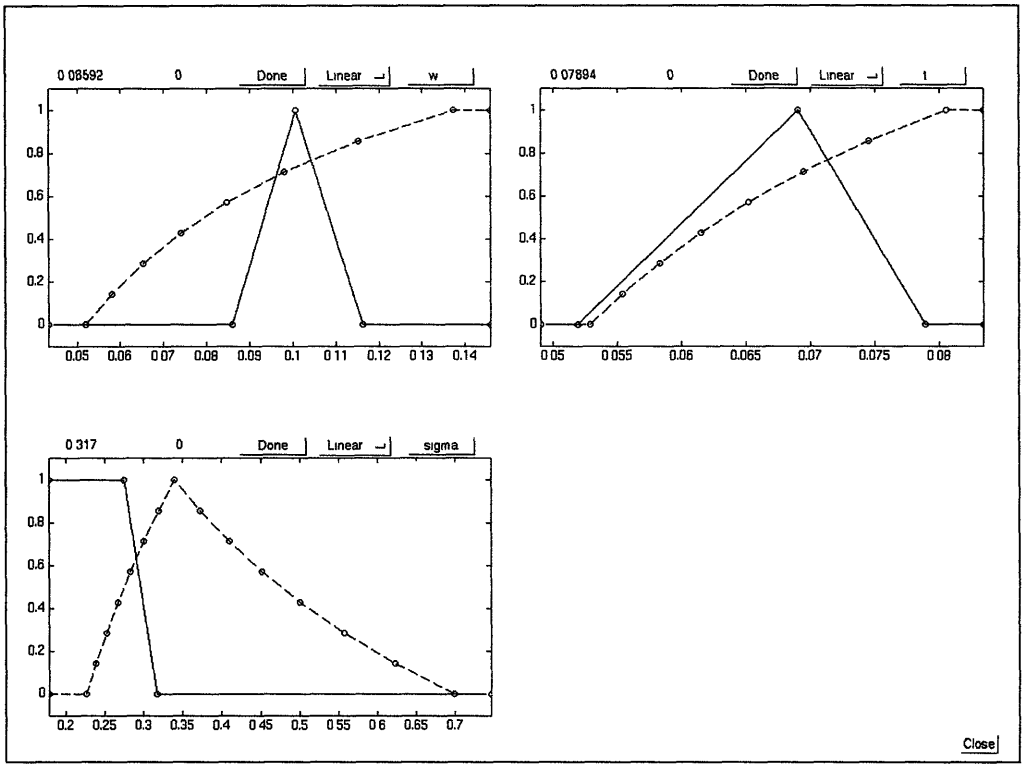

Figure 5-14: Scenario 9: At this point, the capacity to change w and t is almost exhausted. The designer then moves the preference curve on σ toward a smaller range of values. The relaxed constraint is re-tightened.

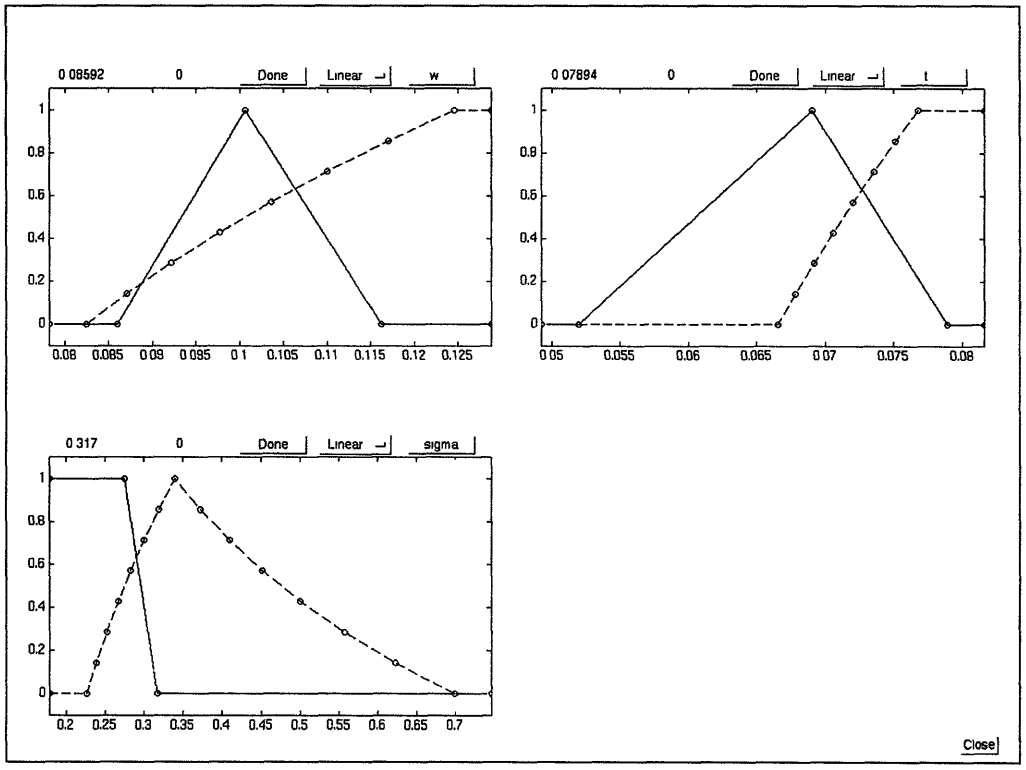

Figure 5-15: Scenario 10: The re-tightened constraint is back-propagated onto w and t.

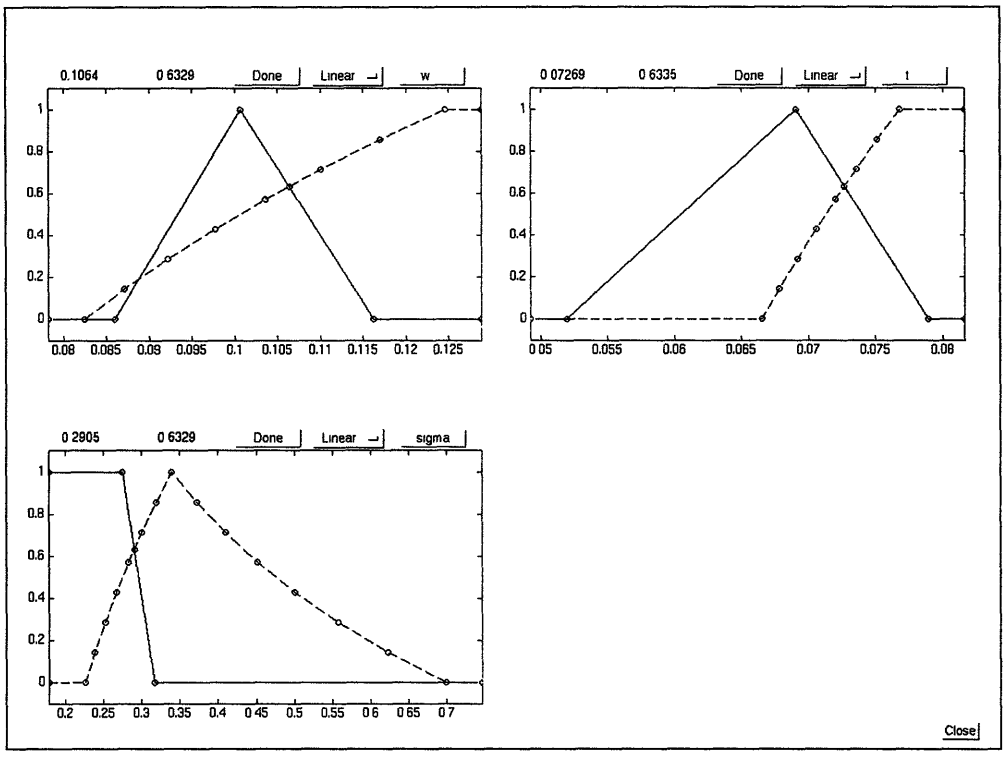

Figure 5-16: Scenario 11: The designer is satisfied with the results. The numerical value of the solution is shown in the two text areas above the left side of each graphical axis.

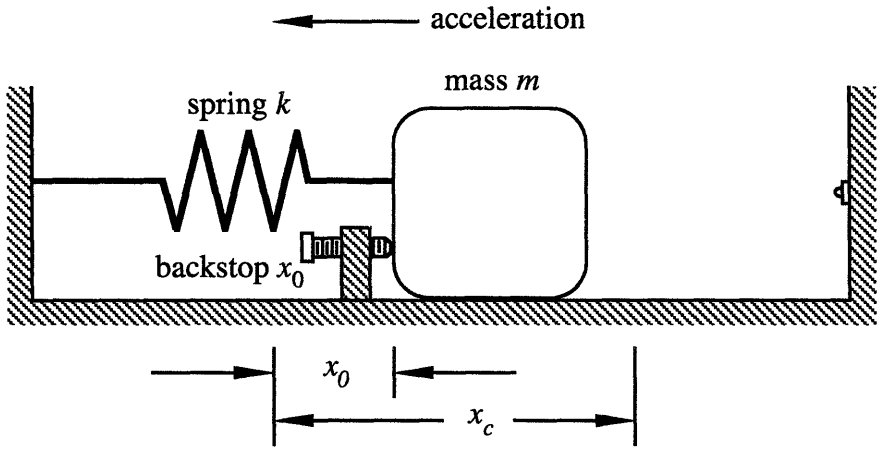

Figure 5-17: Accelerometer design.

5.3 Accelerometer Design

Consider the design of a uni-directional accelerometer which indicates accelerations above a threshold with a switch closure, as first introduced in [40]. In the accelerometer design shown in Figure 5-17[1], there is a mass M attached to a spring k attached to the ground. The ground is accelerated. With sufficient acceleration, the mass must displace a specified distance to make contact with a switch. There is also a backstop placed against the mass, so that the spring k pulls against pre-load P under no acceleration.

Under specified accelerations, the accelerometer mass must contact a switch within specified time durations. However, suppose the spring is a piece of sheet stock formed by a stamping procedure. The inaccuracies introduced by the stamping, attachment and assembly manifest themselves as random errors on k, the spring constant. This uncertainty occurs randomly. Hence, due to the manufacturing process, it is difficult

[1]Taken from reference [33].

to set precise actuation times (time for the mass to touch the actuation switch). We would like to see in what specification ranges would the design variables be to ensure the performance requirement.

There are two goals in this design: to maintain a specified pre-load P, and to close the switch in time τ under a specified acceleration. The variable τ reflects the desired actuation time, and the pre-load P reflects the desired insensitivity to weak accelerations. As a part of these goals, the designer needs to determine whether the design can be made sufficiently tolerant to variational noise to satisfy the customer.

There are two design variables, mass M and spring constant K. There is, however, uncertainty in the manufacture of the spring: a random variation on K, denoted δk. Finally, to assist in maintaining the targets on the goals, the manufacturing procedure can position the backstop based on measurements made of the total spring constant ($k = K + \delta k$) of each accelerometer. This backstop distance is denoted x_0, and is a tuning parameter. The switch distance is denoted x_c. The position of the mass at any given time is denoted x. The mass is to make contact with the switch when subjected to acceleration a.

To determine the time to actuate the switch, the differential equation of motion of the mass must be solved. It is:

$$M(\frac{d^2x}{dt^2} + a) \times H(x - x_0) + kx = P \times H(x_0 - x) \tag{5.2}$$

where H is a step function, $x(0) = x_0$, and $\dot{x}(0) = 0$. This can be solved for the time to actuation:

$$\tau = \sqrt{\frac{M}{k}} \times \arccos\left(\frac{Ma - k(x_0 - x_c)}{Ma}\right) \tag{5.3}$$

This solution assumes, of course, a is sufficiently large to move the mass (i.e., the *arccos* is defined). The other goal is the pre-load P, whose equation is also determined from the above differential equation:

$$P = kx_0 \tag{5.4}$$

Table 5.3: Specified design variable preferences for accelerometer.

Variable	Low end	Optimal	High end	Unites
x_0	0.005	0.007	0.009	m
M	0.0125	0.015	0.0175	Kg
k	1.625	2.0	2.375	N/m
τ	0.0085	0.0095	0.0105	s

Table 5.4: Accelerometer: Constant values.

Variable	Value
a	200 $\frac{m}{s^2}$
x_c	0 m

Maintaining a specific pre-load helps eliminate spurious switch closures.

Having formulated the problem, we can now solve it with the ICPT tool (shown in Figure 5-19 to 5-22). The starting design specification is as in Table 5.3 and shown in Figure 5-18.

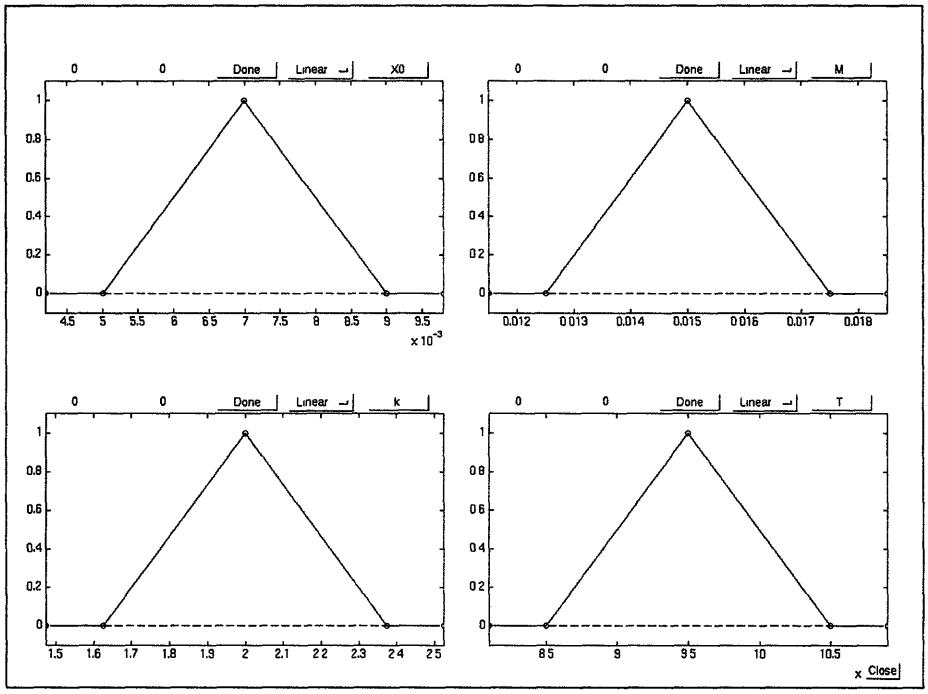

Figure 5-18: Accelerometer: M, K, P, and τ preferences..

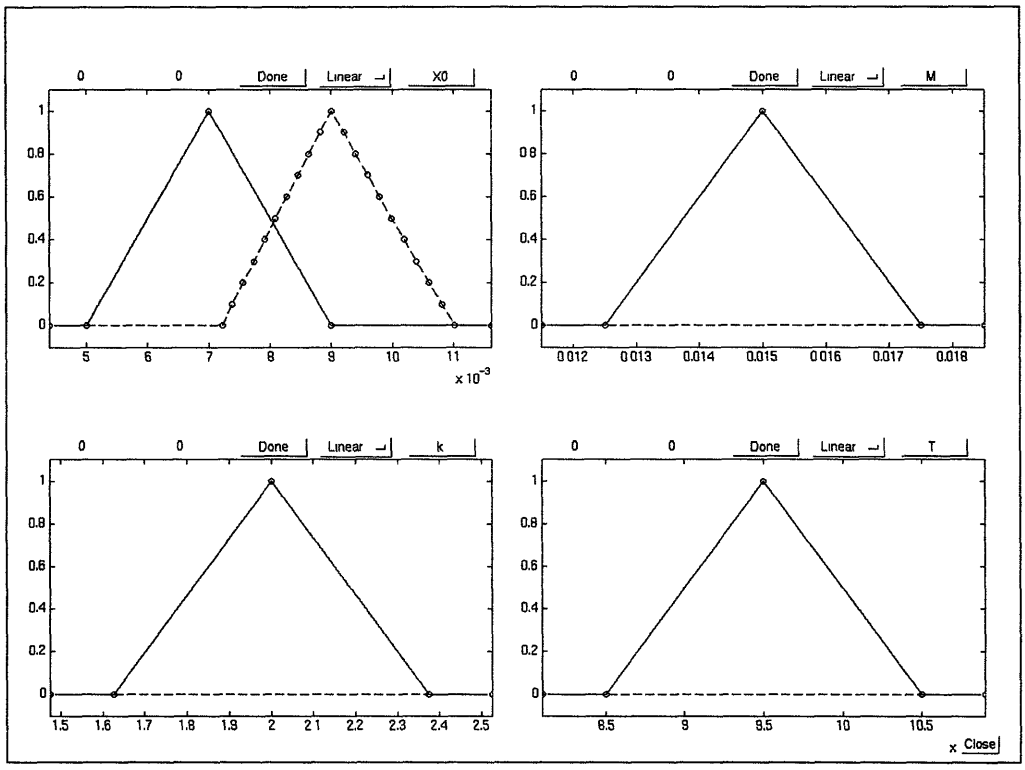

Figure 5-19: Accelerometer: The preference is propagated onto x_0.

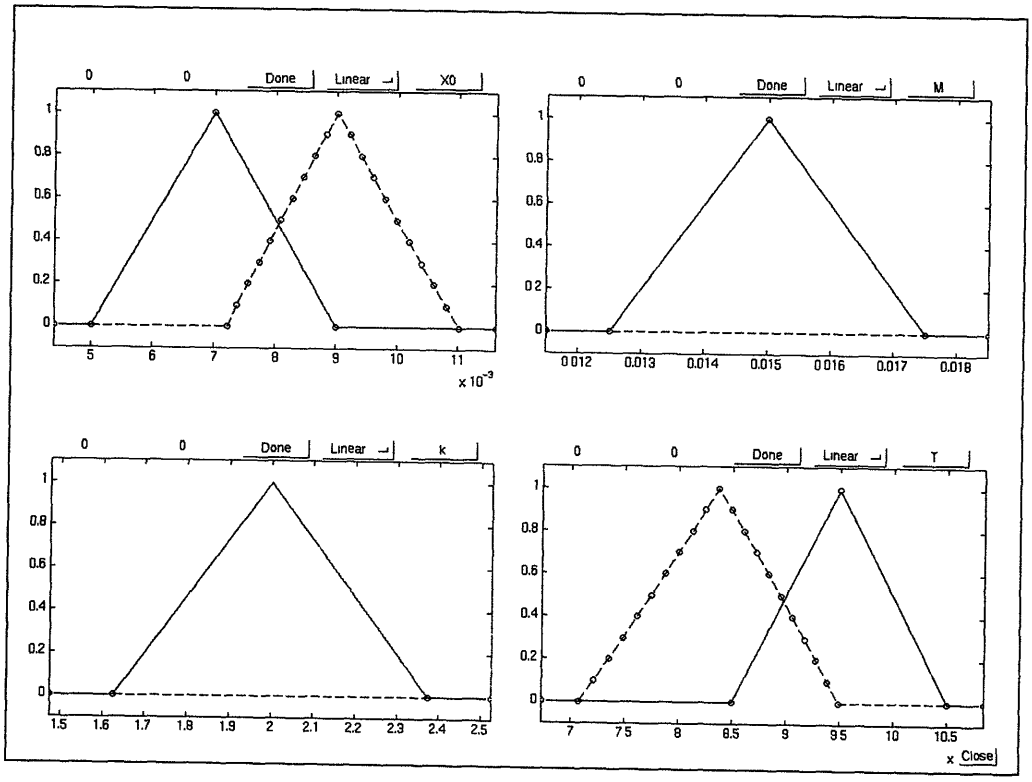

Figure 5-20: Accelerometer: The preference is propagated onto T. The tool fails to converge on the propagations onto M and k; this is due to the insensitivity of T to M and k. The derivative matrix is almost singular. See section in Chapter 6 about dimensional analysis.

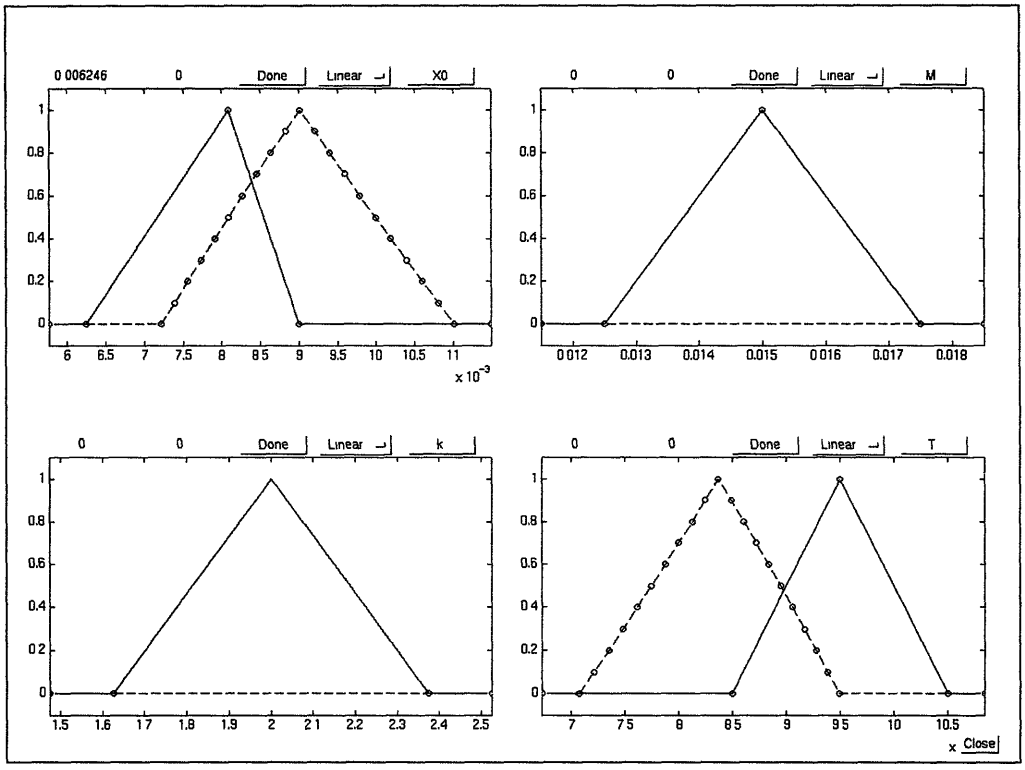

Figure 5-21: Accelerometer: The preference curve on x_0 is modified.

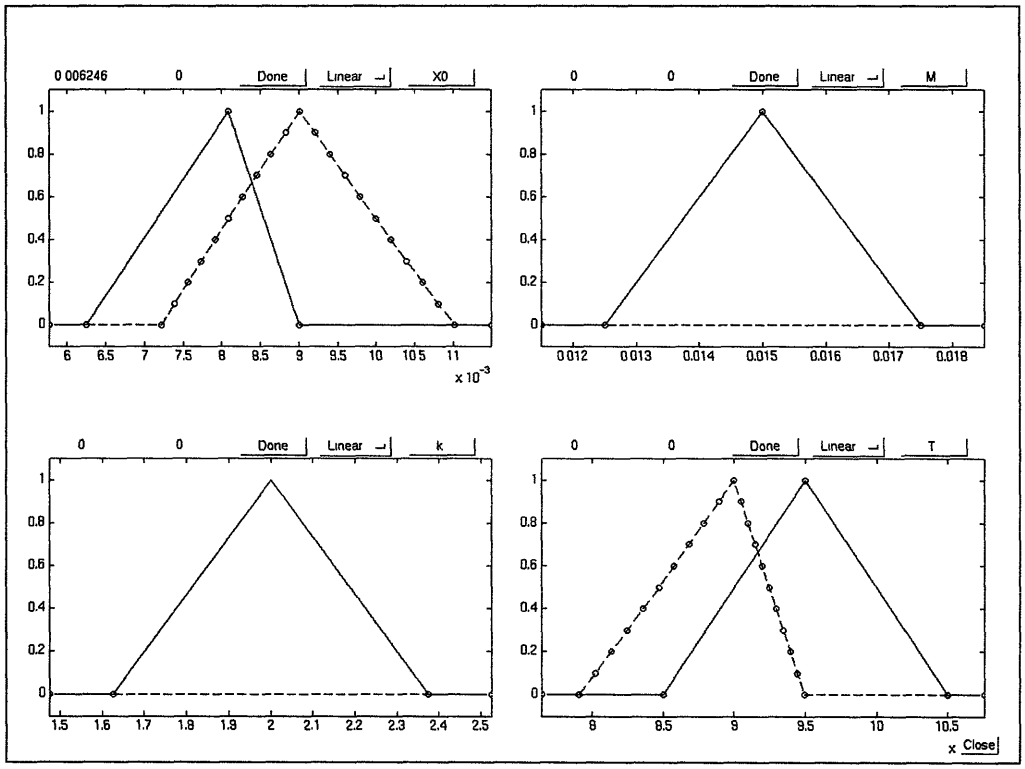

Figure 5-22: Accelerometer: Propagation to T.

Chapter 6

Future Development

6.1 Introduction

This work has two main directions for future development: one in concurrent engineering and the other in robust design.

The design of most complex artifacts requires combining the expertise of specialists in several discrete areas. The various kinds of expertise involved in generating the design of a complex artifact can be employed in either a "sequential" or a "concurrent" mode. In sequential design, the design task is broken down into a sequence of design subtasks for which the output of one designer's decision process serves as input to another. In concurrent design, distributed problem-solving by specialists from different disciplines (i.e., process, structural, mechanical, electrical) or functions (planners, designers and manufacturers) occurs simultaneously, and design decisions must be either coordinated in real time or reviewed for consistency and modified as needed during periodic reviews. Routine or semi-custom design can often be done sequentially, because experience has produced a feasible sequence of design decisions that can be executed serially and without much backtracking. The design of complex or substantially innovative artifacts typically requires a significant degree of concurrent decision-making by its designers. I propose to develop computer tools to assist concurrent engineering decisions. The design task can first be decomposed into subtasks [22] based on the "strength" of interactions of constraints. Strongly coupled

constraints are grouped within a single design group. The interactions among each design group are through *shared variables* [8]: i.e., by putting preference specifications on them. A negotiation strategy will be developed to reconcile conflicting specifications by different groups. When no satisfying solutions are found, we also want the system to tell us what design parameters or constraints are causing most of the troubles and suggest subsequent development of new concepts.

In preliminary design, dominant uncertainties such as inexactness, ill-definedness and vagueness, are invariably non-stochastic. Examples of such imprecision are exactness of concepts, correctness of statements and judgements, which have little to do with the occurrence of events. However, for design problems in which tolerance specifications are important to both performance and cost, the consideration of stochastic imprecision becomes equally important. Another stochastic uncertainty is the tuning parameter [33], which is essentially manufacturing adjustment used to reduce product variations. To model tuning parameters and tolerances one must deploy probabilistic theories. One can achieve product robustness through choosing the right tolerance and tuning adjustment, but in the mean time one also wish to maximize the overall preference of the whole design, under non-stochastic uncertainties. Thus, a research direction could be incorporating both stochastic and non-stochastic uncertainties into preliminary design, and implementing it into a computer tool.

6.2 Tool for Concurrent Engineering

The implementation of concurrent engineering in the U.S. has been through organizational design, creating highly structured design processes and multi-functional teams [30]. However, at Toyota Motor Company, the most successful of the Japanese automotive companies, the key element of concurrent engineering is *set-based design* [58]. When using a set-based approach, designers explicitly communicate and reason about sets of design alternatives. These sets are gradually narrowed down through the elimination of inferior alternatives until only the final solution remains. This approach contrasts with the common practice of iteration (i.e., making several

modifications or improvements in a series) on one alternative until a satisfactory solution emerges. The fuzzy-set approach was originally developed for preliminary design decision-making. It can also be applied to computer-aided set-based concurrent engineering since it manipulates and propagates sets of design parameters and constraints. Furthermore, preferences are specified on the set, thus providing the extra structure to enable additional operations on the set. This feature is especially valuable when making trade-off decisions, in which ranking of different design alternatives is essential for comparisons.

I propose the following subtask and major steps in a computer-aided concurrent engineering design environment.

6.2.1 Establish Functional Blocks and Sub-Systems

Pahl and Beitz's design theory [46] suggests that one of the foremost steps in conceptual design is establishing functional structures of the product. Design teams are then assigned to realize each function. For example, modern automobile engineering divisions consist of climate control, electrical & fuel handling, engine operations & powertrain, plastics & trim products, transmission & chassisline, etc. These relatively independent sub-systems perform functions that comprise the engineering body of a car. Typically, groups of specialized engineers are assigned to work in these different areas. Working on separate functional structures not only brings out the best of the expertise of the design engineers, but also reduces the complexity of the design problem and facilitates team effort. Yet there need to be interactions between these divisions. For example, engine should be compatible with power train; engine size and thrust should be reflected on the locations and strength of the engine mount on the chassis. The problem is how to communicate effectively.

6.2.2 Identify Local Variables and Shared Variables within Each Sub-System

The underlying criteria for establishing boundaries among functional blocks and

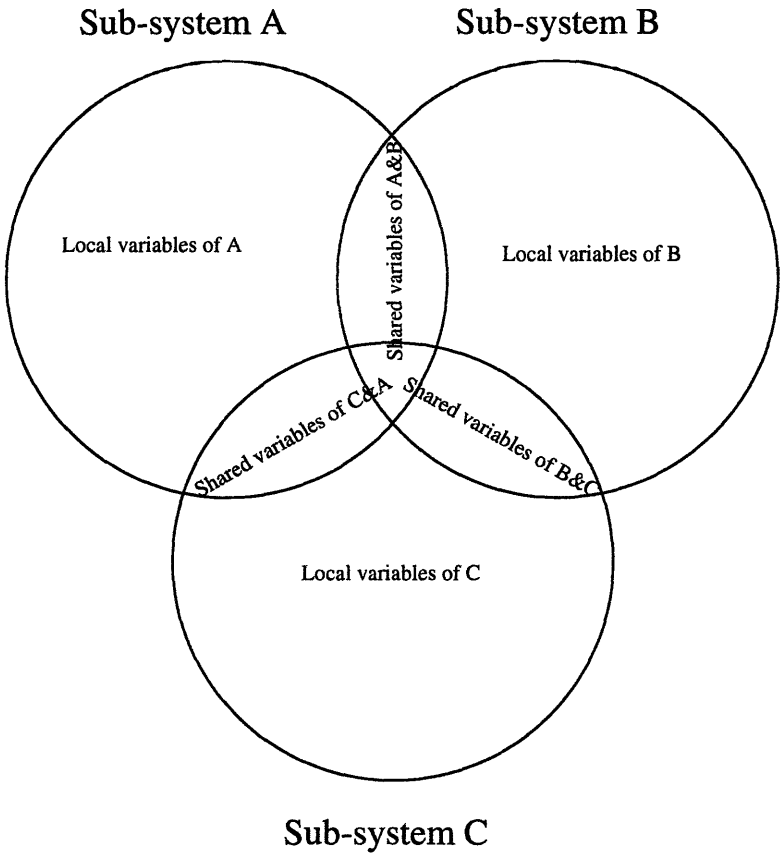

Figure 6-1: Local variables and shared variables among three sub-systems.

design teams is the intensity of the interactions among variable components of the product. Strongly coupled components are grouped together and modeled using *local variables* [8] within a sub-system, with a net of constraints to be satisfied simultaneously or at least satisfactorily. This modeling scheme will enable constraint-based design without much consideration of other loosely coupled constraints. A small but strongly-coupled set of constraints defines a workable design space; on the other hand, it ignores the constraints and design intents that other design teams might impose on the sub-system. Quantitatively, we can model the interactions of sub-systems through *shared variables*, which are variables shared by at least two sub-systems. Consider the design of a front-wheel suspension system. Suppose the design team decides to use a spatial four-bar linkage; then local variables such as the control-arm profile and dimensions, bushing specifications and coil spring constant, etc., can all be specified within the team. For the control-arm to be installed on the chassis, the span of the control arm joint must be communicated to the chassis manufacturer so that a proper set of holes are drilled to house the control arm. Thus, this dimension is shared by both sub-systems.

6.2.3 Setting Feasible Ranges on Local and Shared Variables

Once the sub-system boundaries, local parameters and shared parameters are identified (that means the conceptual design becomes firm), each design team is responsible to achieve their local optimum solutions, subjected to local objectives. However, other loosely-coupled constraints such as manufacturing constraints, cost and interface problems, have to be considered by setting feasible regions on the local parameters, and more importantly, on the shared parameters. For example, at Toyota, the manufacturing constraints are communicated through "lessons learned" books. A lessons learned book for the design of a fender is about "10 to 12 pages long, and contains approximately 60-72 different key ranges of specifications that would ensure the manufacturability of fender design (e.g., intervals of acceptable radii of curvature

for key bends)" [58]. Setting feasible ranges on shared parameters will refine the design space as a whole; in the meantime it insures that sub-system designs agree with each other to an acceptable degree, by defining what can and cannot be done from the point of view of each functional area.

6.2.4 Representing Quality Loss

Taguchi defines "quality loss" as "the loss a product causes to society after being shipped, other than any losses caused by its intrinsic function." He believes that the desirability of a product is determined by the societal loss it generates from the time it is shipped to the customer; the smaller the loss, the higher the desirability. Here we extend the idea to the multi-team environment; each team represents a small society, with their own optimal designs, which have 0 quality loss to them. Any deviations imposed on these locally optimal designs will result in a quality loss. Suppose a team is responsible for designing the tooling for a machine part. If the goal of the design is within the manufacture capacity of the firm, the tooling can always be done by modifying the existing machinery: i.e., setting new fixtures and loading different modular parts. But if the design intent is too far from existing technologies, the quality loss of the team is great: they have to work long hours and find innovative solutions; new equipment must be ordered, etc. Defining quality loss is analogous to setting feasible ranges, except that it quantifies the relative desirability on a common metric: the cost related to changes from each group's locally optimum designs.

6.2.5 Trade-Off Strategies

Research has been done on conflict management among cooperating agents [19]. Trade-off strategies [37] have been proposed to allow designers to explicitly rate and trade-off the design alternatives, using a fuzzy set approach. Although this approach reflects the relative preference among same aspect of a design, it lacks the comparison metric to gauge other aspects of the design. Furthermore, operations like addition lack physical foundation on fuzzy sets.

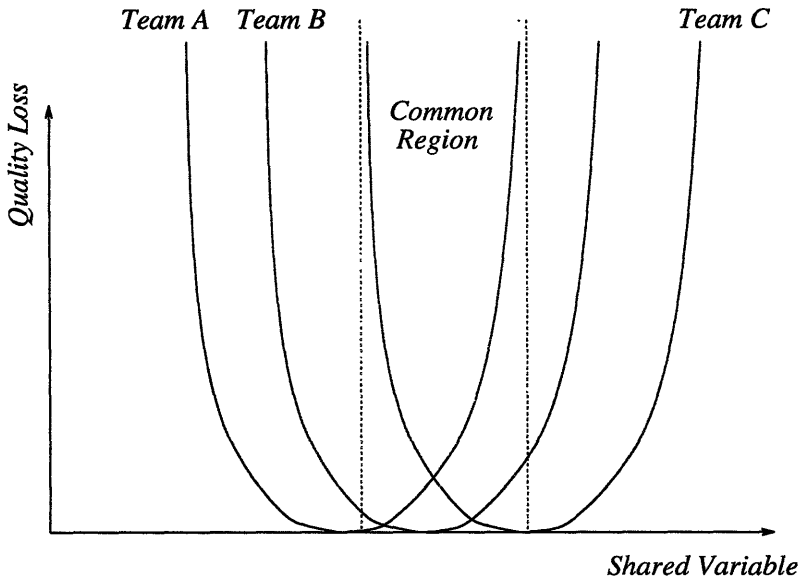

Figure 6-2: Quality loss functions imposed by teams have a common region.

The definition of quality loss attempts to solve the above-mentioned problem and to facilitate the trade-off process. In order for the final product to work, all shared variables have to meet certain degrees of agreement, among every design team who specifies the shared variables. By setting a feasible region for the shared variables, sub-designs agree with each other to an acceptable degree. Some shared variables have to be exactly matched, to ensure constraints like mating conditions for assembly. Some only need to be agreeable in the operational range. To achieve a common value on these variables, design teams must change their designs; thus, deviation occurs. Exactly how much each team should change intuitively depends on the relative difficulty for making such changes. The quality loss of each team will provide a quantitative measure for such difficulty. The optimal trade-off, for example, could be the one that results in minimum total quality losses.

Trade-off works only for shared variables with fairly agreeable specifications by the design teams who share them: i.e., the quality-loss functions imposed by teams have a common region. This is shown in Figure 6-2.

However, sometimes there is no common region to begin with. An example is the

'95 Nissan *Maxima*. The overall goal of its management team was to keep the price below $20,000, yet maintain its reputation as a powerful and slick car. The engine group's original design was not acceptable under this overall goal. To keep the cost low and yet maintain the performance of the engine, they came up with the brand-new, aluminum-alloy V6 engine. It is 23% lighter, and yet delivers 15 additional lb-ft of torque. It is one of the best in its class [20, 31].

In a concurrent engineering environment, innovations are frequently brought into consideration to resolve a conflict of goals. One would like to know what is the most likely place for such innovation to occur. In the *Maxima* example, the engine is an obvious choice since it is the single most costly part of a car. Trade-off decisions made on a quality-loss basis will likely be able to pinpoint conflicting constraints which manifest in the non-overlapping quality-loss functions. Efforts can be directed to improve the most needed aspects of overall design.

6.3 Design with Noises and Tuning Adjustments

Unlike imprecision, other parameter forms remain uncertain throughout a design process. Once identified, the variables that behave probabilistically are not under the control of the designer. A designer can observe the range of a probabilistic variable, decide if it is excessive, and change the process by which the probabilistic uncertainty arose so that the variation is more controlled. For example, a manufacturing process may be altered to allow for tighter fabrication tolerances. Whether tolerance allocation should be considered in the preliminary design stage depends on the importance of tolerance to the performance of the product and the cost associated with it. For design variables with no geometric significance to assembly and function, tolerance is generally not important. An example is the weight of a pendulum for a mechanical clock, since it has nothing to do with its primary function: setting a time unit. The period of the pendulum has nothing to do with the weight of it. However, the tolerances of the gear sets inside it have a large effect on the accuracy and the cost; thus, it must be considered in the early stages of the design. A method has been

developed to model probabilistic noises along with imprecision [34]. Work still needs to be done to implement it into an interactive environment. ICPT has already supported an interactive graphical interface and a constraint maintenance architecture. The difference lies in different operations performed on the imprecise parameters and probabilistic parameters. As shown below, the best performance under imprecision is elicited via an optimization process:

$$d^* : \mu(d^*) = \sup \{\mu(d), d \in \text{DVS}\}$$

where

$$\mu(d) = \mathcal{P}(\mu_1, \ldots, \mu_N)$$

is the overall preference calculated at a point $d \in$ DVS.

For probabilistic parameters, the preferential performance of a point $d \in$ DVS is defined by

$$\mu(d) = \int_{NVS} \mu(d, n)\, dPr$$

where $\mu(d, n) = \mathcal{P}(\mu_1, \ldots, \mu_N)$.

Here the operation is simply integrations. Evaluation of integrals is usually faster than optimization, and is guaranteed to converge. This poses two major advantages when developing a computer-aided tolerancing tool. Industry has substantial interest in such a tool.

Tuning parameters have been thoroughly studied by Otto [40, 33]. Previously called "slack variables," tuning parameters are factory manufacturing adjustments commonly used to correct variational noise errors in a product. Some examples are voltage-supply adjustments, adjustable links and screws, or shims. As an example of a tuning variable, consider the accelerometer example. This product can be modeled as a simple mass-spring system, as shown in Figure 5-17. Under specified accelerations, the accelerometer mass must contact a switch within specified time durations. However, suppose the spring is a piece of sheet stock formed by a stamping procedure. The inaccuracies introduced by the stamping, attachment and assembly manifest them-

selves as random errors on k, the spring constant. This uncertainty occurs randomly. Hence, due to the manufacturing process, it is difficult to set precise actuation times (time for the mass to touch the actuation switch).

A common response to this problem would be to improve the manufacturing process relevant to k, forcing the process to adjust until it exhibits the specified target. If the spring is causing quality problems, then fix the spring. This product has different, cheaper alternatives, however. Specifically, the backstop position might be adjusted to compensate for performance variations caused by the spring. During manufacturing, the spring constant k of every accelerometer could be measured, and the backstop of each accelerometer positioned accordingly to meet the specified actuation times. A potentially equally valid method is to adjust the mass by removing material from it. These are examples of manufacturing adjustments represented by tuning variables.

In addition to this example, tuning variables are also observed in other engineering problems such as automotive carburetor idle positioning, radio or television signal-tuning circuit adjustments, car-seat positioning links, etc. They are characterized by the tuning variable's ability to increase quality and robustness by compensating for noise.

The effect of tuning adjustment can be evaluated using probabilistic methods. Combining tuning parameters and tolerancing will explore opportunities of reduced manufacturing cost by shifting the cost from tight tolerancing to less strict tolerancing but with tuning adjustment. Like most tolerancing methods, this approach minimizes cost while achieving quality goals. Research will be done to construct tuning cost models and tolerancing schemes.

The final goal is to implement robust design methodology into a computer tool. One can envision such a tool which extracts geometric and functional information from a CAD model and interactively changes and shifts probability curves associated with the tolerance specifications, just as we did with the preference curves. The core of the tool will evaluate the expected values of all major performance criteria. The bulk of the research will be on the management of large sets of tolerance chains and functional requirements and on how to interface with a CAD tool such as Pro Engineer. A recent

trend in CAD/CAM software is variational modeling [12], which allows designers to make simultaneous changes to variables, instead of making sequential changes in parametric CAD tool like Pro Engineer. More studies need to be done to find the appropriate CAD/CAM software for this project.

6.4 Some Mathematical Techniques for Design Research

6.4.1 Dimensional Analysis

Dimensional analysis is a method used in the design and analysis of experiments in the physical and engineering sciences. Refer to [5, 56, 57]. When a functional relation between variables is hypothesized, dimensional analysis can be used to check the completeness of the relation and to reduce the number of experimental variables. In modeling problems with known functional relations, functional analysis will provide the most succinct form, give insight about the problems, and save time for robust experimental design.

Taking the accelerometer design again as an example, we want to reduce the dimension of the problem and thus reduce the computation time. With $x_c = 0$, Equation 5.3 becomes

$$\tau = \sqrt{\frac{M}{k}} \times \arccos\left(\frac{Ma - kx_0}{Ma}\right). \tag{6.1}$$

From Equation 6.1 we conclude that

$$\tau = f(M, k, a, x_0) \tag{6.2}$$

We choose M, k, a as independent variables. Then τ, x_0 can be written as:

$$x_0^* = x_0 \frac{k}{Ma}$$

$$\tau^* = \tau\sqrt{\frac{k}{M}}.$$

The dimensionless form of Equation 5.3 should be:

$$\tau^* = f(x_0^*),$$

or

$$\tau^* = \arccos(1 - x_0^*). \tag{6.3}$$

Applying dimensional analysis here not only reduces the design variables from 3 to 1, but also demonstrates possible redundant design variables. From the definition of x_0^* and τ^* one can see clearly that M and k appear only in the form of M/k. This indicates that one of M, k is redundant, only the ratio of them is truly independent. However, such observations are hard to make without dimensional analysis. In experiment design, reducing the number of independent variables means time and experiments saved in many folds. Obtaining a set of truly independent variables is also important for optimization. Since redundant variables tend to waste machine time on trivial circumscriptions; the optimum found is not unique either. Another advantage of performing dimensional analysis is to find a better understanding of functional relations. In the accelerometer example, one may find that τ is relatively insensitive to k and M, but very sensitive to x_0. Once may notice that with the given ranges of values, $x_0^* \ll 1$. Thus, the usual approximation $cos(x) = 1 - x^2/2$ applies. Equation 6.3 becomes

$$\tau^* \simeq \sqrt{2x_0^*}.$$

Thus

$$\frac{\tau}{\sqrt{M/k}} \simeq \sqrt{2\frac{x_0}{Ma/k}}$$

or

$$\tau \simeq \sqrt{2x_0/a}.$$

This explains the insensitivity of τ to M and k, and the difficulties in Figure 5-20.

6.4.2 Sensitivity Analysis

Although the LIA algorithm is applicable in a range of engineering problems, its inherent disadvantage is that the number of calculations grows exponentially with the increase in the number of independent variables being solved.

Using dimensional analysis may reduce the calculations by many times, but it is still unacceptable for large systems with many independent variables. This adversely affects the interactivity of ICPT. Sensitivity analysis has the potential to greatly reduce the number of function evaluations. Once the sensitivity of a performance variable respective to an independent variable is known, that independent variable can be taken out of experimental-design space. The assumption made here is merely monotonicity, the same used in LIA and ELIA. Although this puts restrictions on applications, it is applicable in the usual cases of engineering design problems where the intervals of design variables are small (which is generally true when the design is approaching completion).

Chapter 7

Conclusion

A constraint-based system is one in which preliminary parametric design has been designed and implemented. For over-constrained systems, *imprecise slackening* relaxes the constraint values and allows the system to be imprecisely constrained. However, a user also has the option to re-tighten the constraints. The *set* of values that satisfies the constraints are found by intersecting the preference and *induced* preference curves. Thus, design space is thoroughly and automatically explored. For systems with multiple constraints, designer can explore the design space beyond the crisp boundary and find better possible designs by trading-off two constraints, thus avoiding committing to constraint values too early in the design stage.

There are limitations in the system. Some lie in the limited capabilities of the nonlinear equation-solver. Both ELIA and the optimization approach may fail to converge, due to the fact that the solver is sensitive to the initial values. Various numerical techniques such as scaling can partially correct these problems. The solver only returns one solution, even though the system could have multiple solutions. This may create problems for periodical functions (such as in the accelerometer example) and polynomial equations (such as a quadratic equation in the truss example). The system has no intelligence about what roots to use, and sometimes fails to find the physical correct solution, such as finding a negative length. Better constraint management and user-interface could be used. However, these are not the focal points of this thesis.

This thesis makes contributions to software tools in design. It allows interactive modification and evaluation of design specifications. Over-constrained systems are relaxed and can be re-tightened once the design space is thoroughly explored. Trade-off of conflicting constraints can be done. The interactive user-interface can be adopted into other design tools, such as tolerance design. The same methodology can be extended to collaborative design and concurrent engineering, provided suitable trade-off metrics can be found.

Appendix A

User Interface Reference

The purpose of this Appendix is to present documentation on the details of the user interface of ICPT. Screen displays will be mostly self-explanatory.

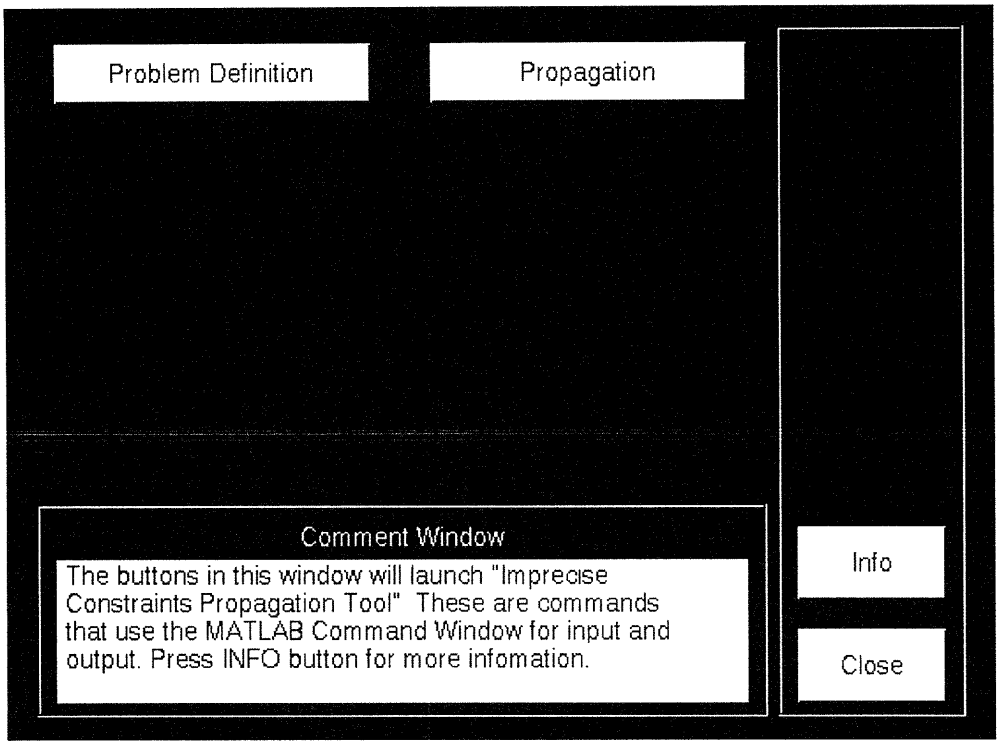

Figure A-1: ICPT command window.

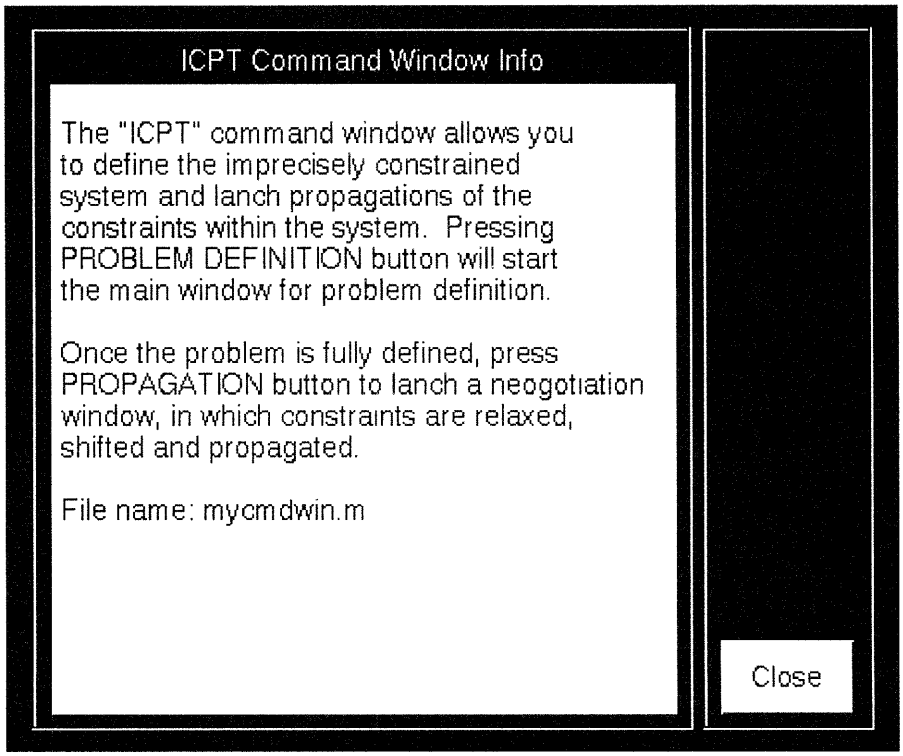

Figure A-2: ICPT command window information.

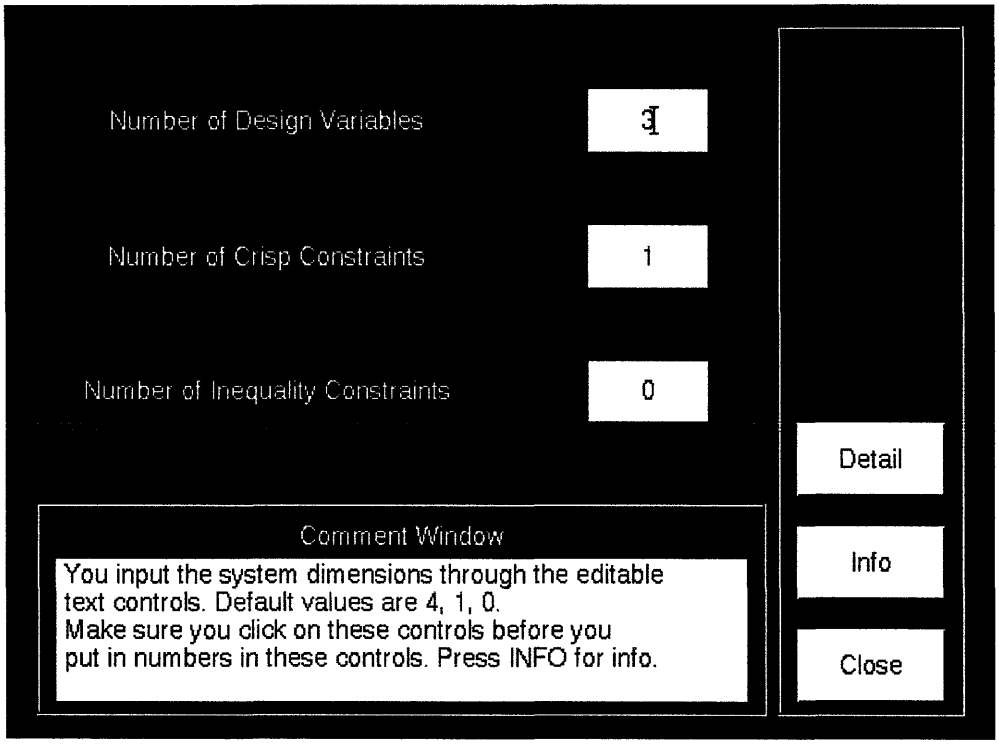

Figure A-3: ICPT main problem definition window.

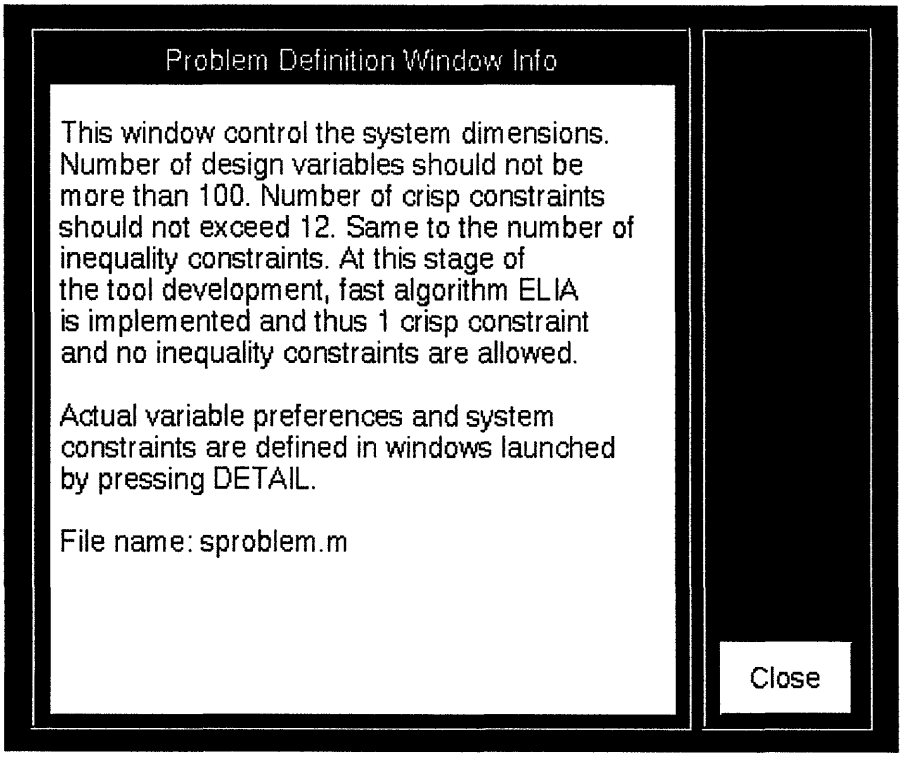

Figure A-4: ICPT main problem definition window information.

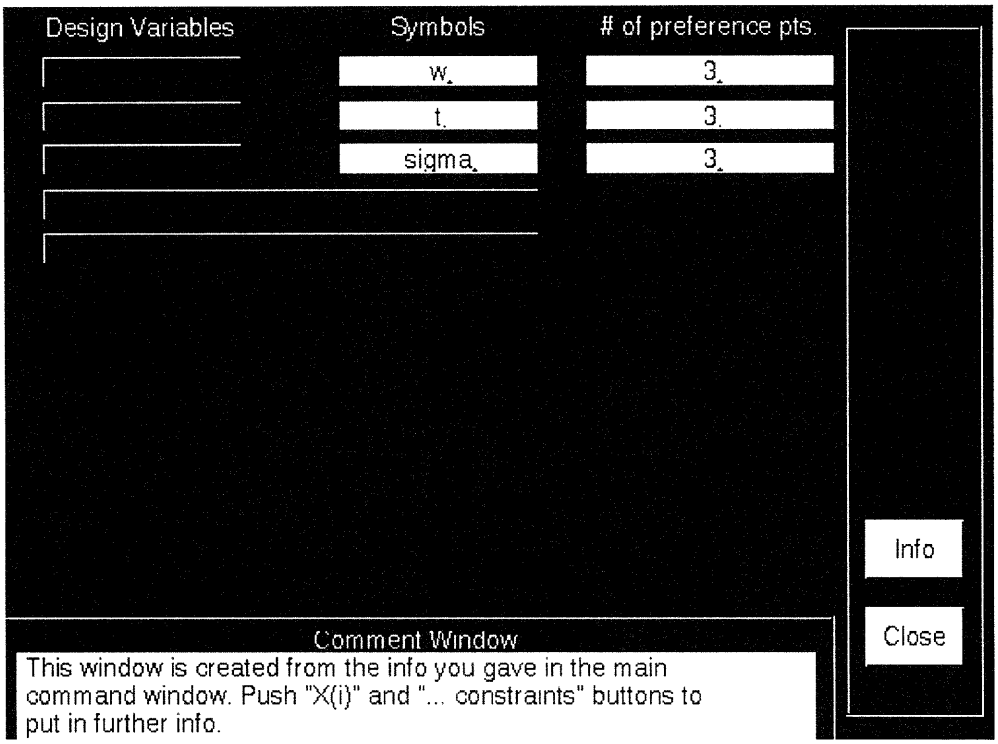

Figure A-5: ICPT problem detail definition window.

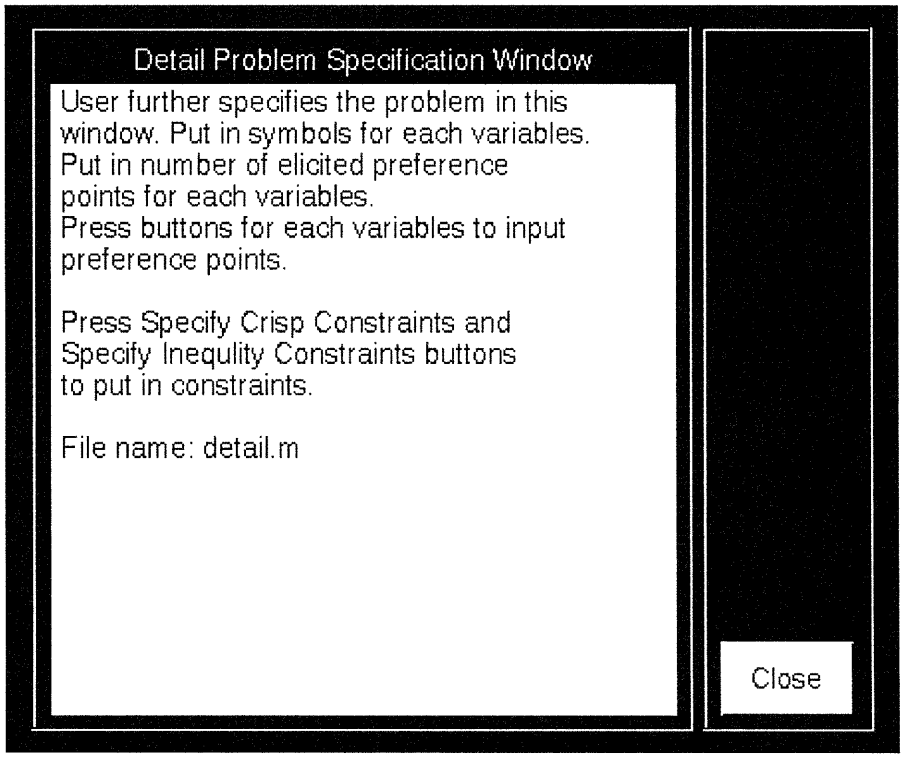

Figure A-6: ICPT problem detail definition window information.

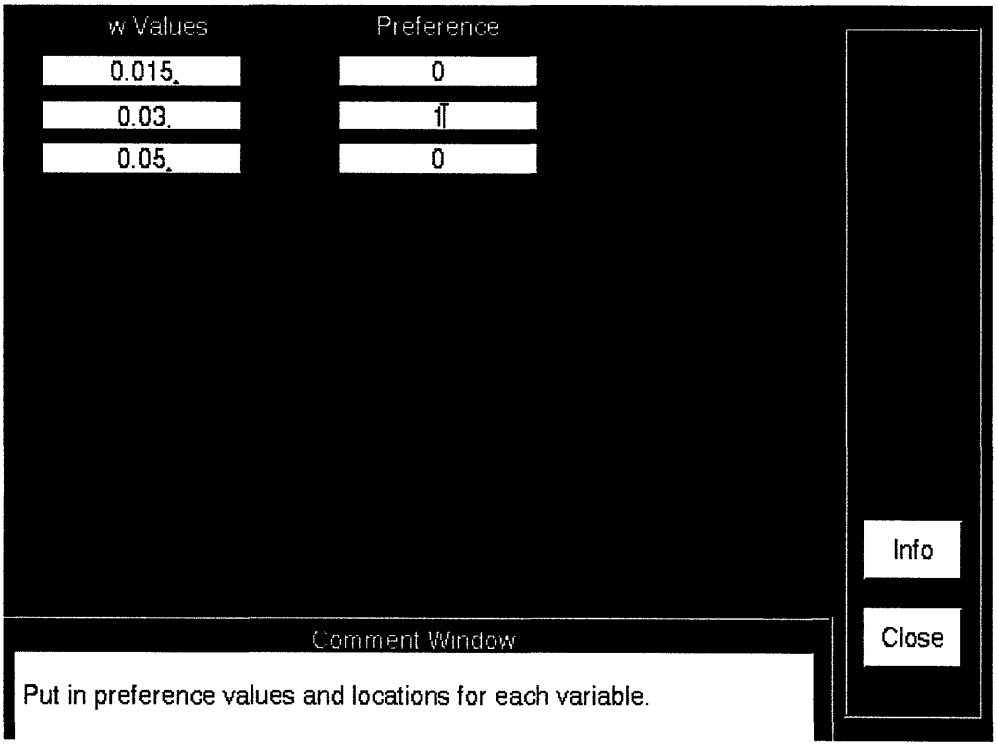

Figure A-7: ICPT preference definition window.

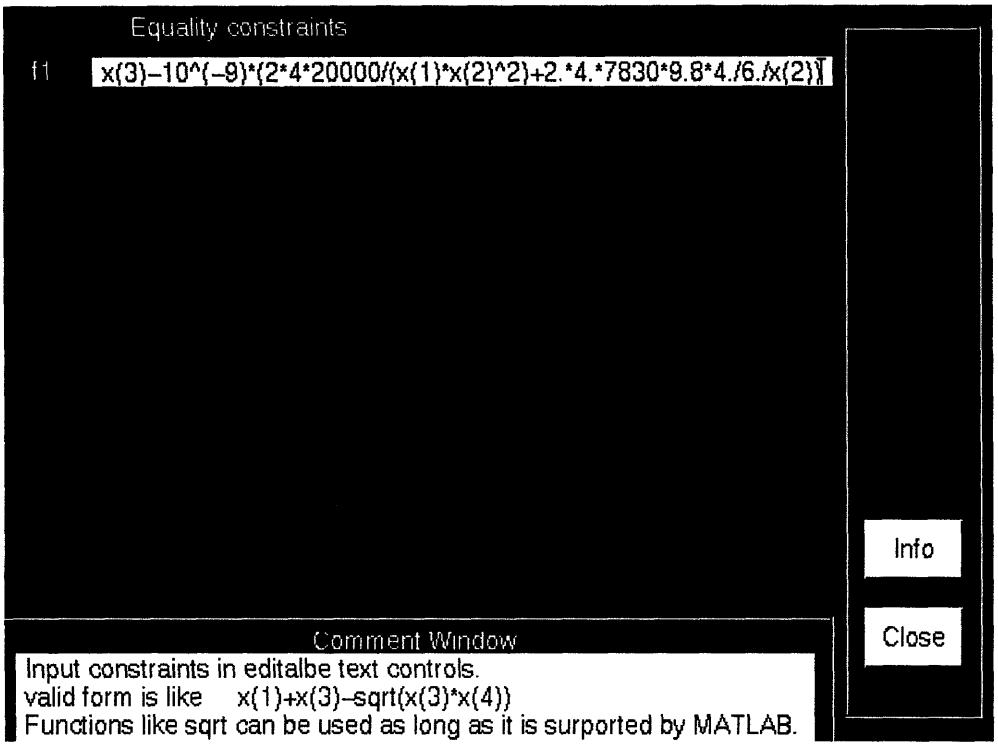

Figure A-8: ICPT crisp constraints definition window.

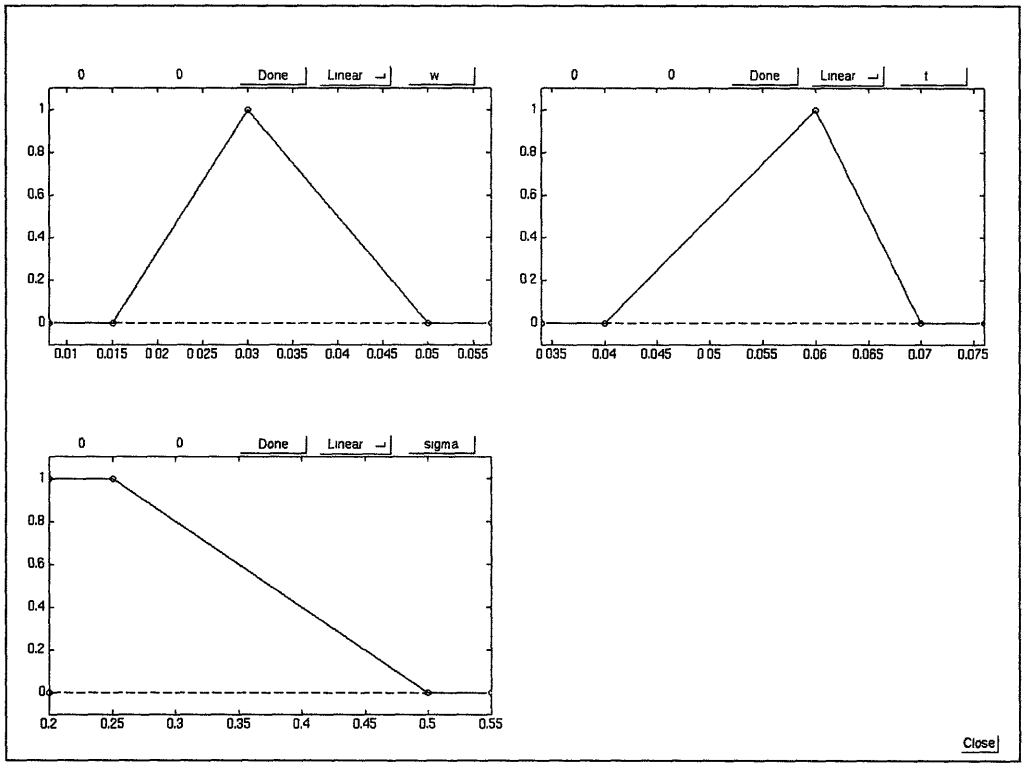

Figure A-9: ICPT constraint propagation window.

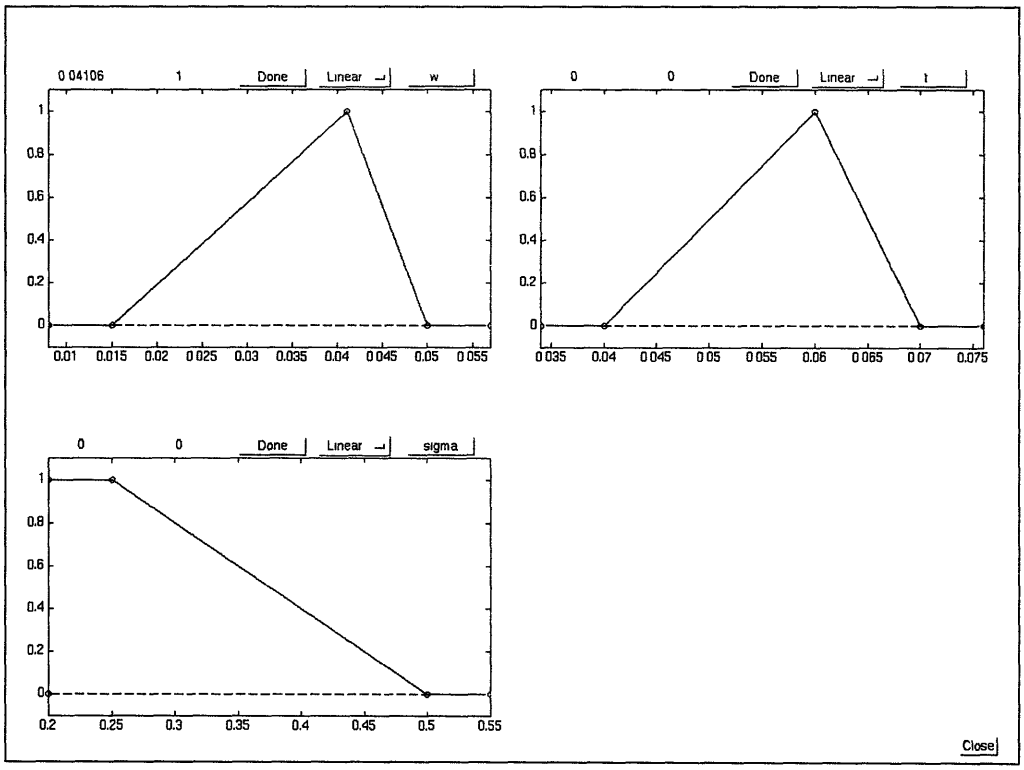

Figure A-10: Preference curves can be modified by dragging the control points around. The text area shows the current position of the modified point.

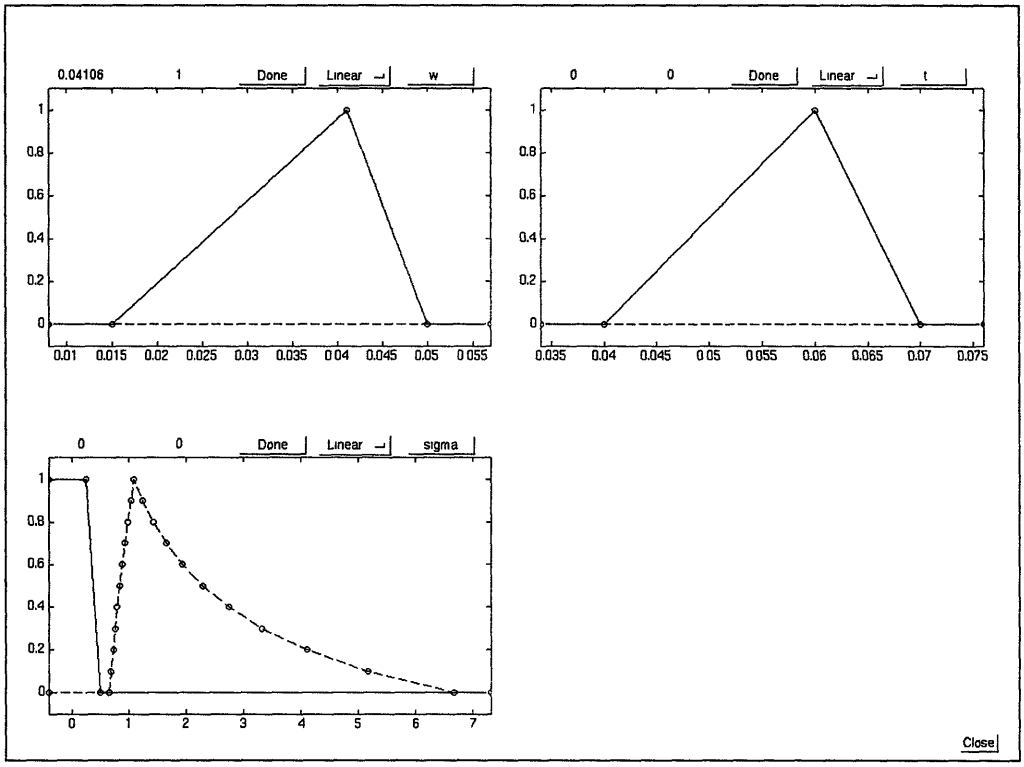

Figure A-11: The modified preference can then be propagated onto other variables by pressing the control button above the right side of each graphical axis.

Bibliography

[1] R. Agrawal, G. Kinzel, R. Srinivasn, and K. Ishii. Engineering constraint management based on an occurance matrix approach. *ASME Journal of Mechanical Design*, 115(1):103–109, 1993.

[2] Y. Akao. *Quality Function Deployment: Integrating Customer Requirements into Product Design.* Productivity Press, Cambridge, 1990.

[3] A. Asgharzadeh, M. Jamshidi, and N. Vadiee. A truth qualified fuzzy logic rule based expert system with application to image enhancement. In *Proceedings of the 1994 FUZZ-IEEE Conference*, pages 340–251, Orlando, FL, June 1994. IEEE.

[4] M. Asimow. *Introduction to Design.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1962.

[5] P. W. Bridgeman. *Dimensional Analysis.* Yale University Press, New Haven, 1931.

[6] J. Chen and K. Otto. A tool for imprecise calculations in engineering systems. In *Proceedings of the Fourth International Conference on Fuzzy Systems*, Yokohama, JP, 1995. Accepted for Publication.

[7] J. E. Chen and K. N. Otto. Constructing membership functions using interpolation and measurement theory. *Fuzzy Sets and Systems*, 1994. Accepted for publication. Engineering Design Reseach Laboratory TR 93-4, Massachusetts Institute of Technology.

[8] Joseph E. Chen. Methods of innovative problem-solving for design and manufacture. *A Proposal for Alfred Keil Fellowship for the Wiser Uses of Science and Technology*, 1994.

[9] A. K. Cline. Scalar- and planar-valued curve fitting using splines under tension. *Communications of the ACM*, 17(4), April 1974.

[10] A. R. Diaz. Fuzzy set based models in design optimization. In S. S. Rao, editor, *Advances in Design Automation - 1988*, volume DE-14, pages 477–485, New York, September 1988. ASME.

[11] A. R. Diaz. A strategy for optimal design of hierarchical systems using fuzzy sets. In J. R. Dixon, editor, *The 1989 NSF Engineering Design Research Conference*, pages 537–547, College of Engineering, University of Massachusetts, Amherst, June 1989. NSF.

[12] Sidney Hill. Modeling techniques spark fierce competition in cad/cam. *Manufacturing Systems*, September 1994.

[13] The MATH WORKS Inc. *MATLAB Reference Guide.* The MATH WORKS, Inc., Natick, Massachusetts, 1992.

[14] The MATH WORKS Inc. *MATLAB User's Guide.* The MATH WORKS, Inc., Natick, Massachusetts, 1992.

[15] The MATH WORKS Inc. *Building a Graphical User Interface.* The MATH WORKS, Inc., Natick, Massachusetts, 1993.

[16] The MATH WORKS Inc. *External Interface Guide.* The MATH WORKS, Inc., Natick, Massachusetts, 1993.

[17] The MATH WORKS Inc. *Optimization Toolbox User's Guide.* The MATH WORKS, Inc., Natick, Massachusetts, 1994.

[18] John R. Jauser and Don Clausing. The house of quality. *Harvard Business Review*, May 1988.

[19] Mark Klein. Detecting and resolving conflicts among cooperating human-and-machine-based design agents. *Artificial Intelligence in Engineering*, 7, 1992.

[20] Gerry Kobe. Nissan maxima. *Automotive Industries*, October 1994.

[21] D. Krantz, R. Luce, P. Suppes, and A. Tversky. *Foundations of Measurement*, volume I. Academic Press, New York, 1971.

[22] A. Kusiak and J. Wang. Decomposition of the design process. *Journal of Mathematical Analysis and Applications*, 115, December 1993.

[23] William S. Law and Eric K. Antonsson. Including imprecision in engineering design calculations. In *Proccedings of the 1994 Design Theory and Methodology Conference*, pages 109–114, New York, 1994. ASME.

[24] W. Lewis and A. Samuel. *Fundamentals of Engineering Design: Ideas, Methods, and Applications.* Prentice-Hall, 1989.

[25] M. Luhandjula. Fuzzy optimization: An appraisal. *Fuzzy Sets and Systems*, 30:257–282, 1989.

[26] David F. McAllister and John A. Roulier. Algorithm 574, shape-preserving osculatory quadratic splines. *ACM Transactions on Mathematical Software*, 7(3), September 1981.

[27] David F. McAllister and John A. Roulier. An algorithm for computing a shape-preserving osculatory quadratic spline. *ACM Transactions on Mathematical Software*, 7(3), September 1981.

[28] K. Muller and M. Tharigen. Applications of fuzzy hierarchies and fuzzy madm methods to innovative system design. In *Proceedings of the 1994 FUZZ-IEEE Conference*, pages 340–251, Orlando, FL, June 1994. IEEE.

[29] D. Navinchandra and D. H. Marks. Design exploration through constraint relaxation. In J. S. Gero, editor, *Expert Systems in Computer-Aided Design*, pages 481–509, Amsterdam, 1987. Elsevier Science Publishers B. V.

[30] J. L. Nevins and et al. D. E. Whitney. *1989 Concurrent Design of Products and Processes.* McGraw-Hill Publishing Co., New York, 1989.

[31] US News and World Report. An article about car business. *US News and World Report*, 1994.

[32] K. Otto. Measurement methods for product evaluation. *Research in Engineering Design*, 1994. Accepted for publication.

[33] K. Otto. Robust product design using manufacturing adjustments. In *Proceedings of the 1994 ASME Design Theory and Methodology Conference*, pages 1–8, Minneapolis, MN, 1994. ASME.

[34] K. N. Otto. *A Formal Representational Theory for Engineering Design.* PhD thesis, Calnifornia Institute of Technology, 1992.

[35] K. N. Otto. Measurement foundations for design engineering methods. In *Proceedings of the 1993 ASME Design Theory and Methodology Conference*, pages 157–166, Albuquerque, NM, September 1993. ASME.

[36] K. N. Otto and E. K. Antonsson. Trade-Off Strategies in Engineering Design. *Research in Engineering Design*, 3(2):87–104, 1991.

[37] K. N. Otto and E. K. Antonsson. Trade-Off Strategies in the Solution of Imprecise Design Problems. In T. Terano et al., editors, *Fuzzy Engineering toward Human Friendly Systems: Proceedings of the International Fuzzy Engineering Symposium '91, Volume 1*, pages 422–433, Yokohama Japan, November 1991. LIFE, IFES.

[38] K. N. Otto and E. K. Antonsson. The method of imprecision compared to utility theory for design selection problems. In *Proceedings of the 1993 ASME Design Theory and Methodology Conference*, pages 167–173, Albuquerque, NM, September 1993. ASME.

[39] K. N. Otto and E. K. Antonsson. Propagation of imprecise constraints in engineering systems. Technical Report 93-2, Massachusetts Institute of Technology, Cambridge, MA, 1993.

[40] K. N. Otto and E. K. Antonsson. Tuning Parameters in Engineering Design. *ASME Journal of Mechanical Design*, 115(1):14–19, 1993.

[41] K. N. Otto and E. K. Antonsson. Design Parameter Selection in the Presence of Noise. *Research in Engineering Design*, 1994. Accepted for publication.

[42] K. N. Otto and E. K. Antonsson. Modeling imprecision for product design. In *Proceedings of the 1994 FUZZ-IEEE Conference*, pages 340–251, Orlando, FL, June 1994. IEEE.

[43] K. N. Otto and E. K. Antonsson. Propagating imprecise engineering constraints. In *Proceedings of the Fourth International Conference on Fuzzy Systems*, Yokohama, JP, 1995. Accepted for Publication.

[44] K. N. Otto, A. D. Lewis, and E. K. Antonsson. Determining optimal preference points with dependent variables. *Fuzzy Sets and Systems*, 60(1), November 1993.

[45] K. N. Otto, A. D. Lewis, and E. K. Antonsson. Fuzzy Preference Induced on Manifolds by Vector Fields and Flows. *Fuzzy Sets and Systems*, 1994. Accepted for publication. Engineering Design Reseach Laboratory TR 91f, California Institute of Technology.

[46] G. Pahl and W. Beitz. *Engineering Design.* The Design Council, Springer-Verlag, New York, 1984.

[47] S. Pugh. *Total Design.* Addison-Wesley, New York, 1990.

[48] R. Ramaswamy and K. Ulrich. A designer's spreadsheet. In K. Hight and L. Stauffer, editors, *Design Theory and Methodology – DTM '93*, pages 105–113, New York, September 1993. ASME. Volume DE-53.

[49] S. S. Rao. Description and optimum design of fuzzy mechanical systems. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 109:126–132, March 1987.

[50] S. S. Rao, K. Sundararaju, B. Prakash, and C. Balakrishna. Multiobjective fuzzy optimization techniques for engineering design. *Computers and Structures*, 42(1):37–44, 1992.

[51] M. Sakawa and H. Yano. Interactive decision making for multiobjective nonlinear programming problems with fuzzy parameters. *Fuzzy Sets and Systems*, 29:315–326, 1989.

[52] M. Sakawa and H. Yano. Interactive fuzzy decision making for multiobjective nonlinear programming problems with fuzzy parameters. *Fuzzy Sets and Systems*, 32:245–261, 1989.

[53] M. Sakawa and H. Yano. An interactive fuzzy satisfying method for multiobjective nonlinear programming problems with fuzzy parameters. *Fuzzy Sets and Systems*, 30:221–238, 1989.

[54] M. Sakawa and H. Yano. Feasibility and Pareto optimality for multiobjective nonlinear programming problems with fuzzy parameters. *Fuzzy Sets and Systems*, 43:1–15, 1991.

[55] D. Serrano. *Constraint Management in Conceptual Design.* PhD thesis, MIT, 1987.

[56] A. A. Sonin. Advanced fluid dynamics (2.25) note on dimensional analysis. Technical report, Massachusetts Institute of Technology, Cambridge, MA, 1994.

[57] E. S. Taylor. *Dimensional Analysis for Engineers.* Oxford University Press, London, 1974.

[58] A. C. Ward and et al. Set-based concurrent design and toyota. In *Proceedings of the 1994 ASME Design Theory and Methodology Conference*, pages 79–86, Minneapolis, MN, 1994. ASME.

[59] K. L. Wood. *A Method for Representing and Manipulating Uncertainties in Preliminary Engineering Design.* PhD thesis, California Institute of Technology, Pasadena, CA, 1989.

[60] K. L. Wood and E. K. Antonsson. A Fuzzy Sets Approach to Computational Tools for Preliminary Engineering Design. In S. S. Rao, editor, *Advances in Design Automation, 1987, Volume One: Design Methods, Computer Graphics, and Expert Systems – DE-Vol. 10-1*, pages 263–271, New York, September 1987. ASME. Presented at the *1987 ASME Design Automation Conference*, Boston, MA, September 27-30, 1987.

[61] K. L. Wood and E. K. Antonsson. Computations with Imprecise Parameters in Engineering Design: Application and Example. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, June 1988. Submitted for review. EDRL-TR 88b.

[62] K. L. Wood and E. K. Antonsson. Computations with Imprecise Parameters in Engineering Design: Background and Theory. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 111(4):616–625, December 1989.

[63] K. L. Wood and E. K. Antonsson. Engineering Design Uncertainties. In J.E.A. John, editor, *1989 NSF Engineering Design Research Conference*, University of Massachusetts, Amherst, MA, June 1989. NSF. 1989 NSF Grantee Workshop on Design Theory and Methodology.

[64] K. L. Wood, E. K. Antonsson, and J. L. Beck. Comparing Fuzzy and Probability Calculus for Representing Imprecision in Preliminary Engineering Design. In W. P. Seering, editor, *Design Theory and Methodology - 1989*, New York, 1989. ASME. Presented at the First International ASME *Design Theory and Methodology Conference* (DTM '89), September 18-19, 1989, Montréal, Canada.

[65] K. L. Wood, E. K. Antonsson, and J. L. Beck. Representing Imprecision in Engineering Design – Comparing Fuzzy and Probability Calculus. *Research in Engineering Design*, 1(3/4):187–203, 1990.

[66] K. L. Wood, K. N. Otto, and E. K. Antonsson. Engineering Design Calculations with Fuzzy Parameters. *Fuzzy Sets and Systems*, 52(1):1–20, November 1992.

[67] K. L. Wood, K. N. Otto, and E. K. Antonsson. Engineering Design Calculations with Fuzzy Parameters. In *Fuzzy Sets and Systems* [66], pages 1–20.