

Optimization Methods for Calculating Design Imprecision ^{*†}

William S. Law
Erik K. Antonsson
Engineering Design Research Laboratory
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, California

Abstract

The preliminary design process is characterized by *imprecision*: the vagueness of an incomplete design description. The *Method of Imprecision* uses the mathematics of fuzzy sets to explicitly represent and manipulate imprecise preliminary design information, enabling the designer to explore the space of alternative designs in the context of the designer and customer's preferences among alternatives. This paper introduces new methods to perform Method of Imprecision calculations for general non-monotonic design evaluation functions that address the practical necessity to minimize the number of function evaluations. These methods utilize optimization and experiment design.

Introduction

Evaluation is a key component of preliminary design. Evaluating alternatives early in the design process avoids further investment in inferior alternatives, and can allow more alternatives to be considered. Traditional evaluation tools are of limited value in preliminary design because they require a precise design description. Preliminary design information is characteristically *imprecise*: the design description is vague and indistinct. The designer must consider a cloud of alternative designs. A computational tool that considers individual designs separately provides limited insight into the structure of the cloud of alternatives. A more systematic methodology is required: one that explicitly models imprecision.

The *Method of Imprecision* [1, 2, 3] uses the mathematics of fuzzy sets to represent and manipulate imprecise preliminary design information. The *Imprecise Design Tool* [3], a computational implementation of this methodology, supports preliminary design decisions based on imprecise information, and accepts any design evaluation function as required by the application.

*Manuscript prepared for submission to the *21st ASME Design Automation Conference, 17-21 September 1995*.

†KEYWORDS: Design optimization; Industrial examples, developments and perspectives.

Practical computational tools for design must be resource efficient. They must not demand too much of a designer’s time and attention, and they must produce satisfactory answers without excessive computation. Frequently the cost of evaluating a design is substantial, and the critical measure of computational efficiency is the number of design evaluations required. This paper presents new methods to calculate imprecision for general non-monotonic evaluation functions that address the practical necessity to minimize the number of function evaluations. Many of these methods evolved from discussions with engineers from the Vehicle Structures Computer Aided Design group at the Ford Motor Company, Dearborn, MI, and in particular Thomas Mathai. The authors gratefully acknowledge their contribution.

Definitions and Notation

Design alternatives are described by a collection of *design variables* d_1, \dots, d_n . These variables need not be continuous or even ordinal: the design variable “styling” may have the unordered values “conservative”, “sporty”, and “futuristic”. The set of valid values for d_i is denoted \mathcal{X}_i . The whole set of design variables forms an n vector, \vec{d} , that uniquely identifies each design alternative in the *design variable space* (*DVS*).

Performance variables p_1, \dots, p_q measure aspects of a design’s performance. Each performance variable p_j is defined by a mapping f_j such that $p_j = f_j(\vec{d})$. The mappings f_j can be any calculation or procedure to evaluate the performance of a design, including closed-form equations, computational algorithms, “black box” functions, prototype testing, and market research. The set of valid values for p_j is denoted \mathcal{Y}_j . The set of performance variables for each design alternative forms a q vector, $\vec{p} = \vec{f}(\vec{d})$, that specifies the performance of a design \vec{d} . The *performance variable space* (*PVS*) encompasses all performances \vec{p} .

Imprecise variables may potentially assume any value within a possible range because the designer does not know, *a priori*, the final value that will emerge from the design process. Yet even though the designer is unsure about what value to specify, certain values will be preferred over others. This preference, which may arise objectively (*e.g.*, cost or availability of components) or subjectively (*e.g.*, from experience), is used to quantify the imprecision associated with a design variable. The preference that the designer has for values of the design variable d_i is represented by a preference function on \mathcal{X}_i , termed the *design preference*:

$$\mu_{d_i}(d_i) : \mathcal{X}_i \rightarrow [0, 1] \subset \mathbb{R}.$$

$\mu_{d_i}(d_i)$ quantifies the designer’s preference for values of d_i , and is distinct from the customary membership function in a fuzzy set, which quantifies the extent to which values belong to the set. The preference that a customer has for values of the performance variable p_j is similarly represented by a preference function on \mathcal{Y}_j , termed the *functional requirement*:

$$\mu_{p_j}(p_j) : \mathcal{Y}_j \rightarrow [0, 1] \subset \mathbb{R}.$$

The combined preference of the designer and customer for a particular design \vec{d} is represented by an overall preference $\mu_o(\vec{d})$, which is a function of the design preferences $\mu_{d_i}(d_i)$, and the functional requirements $\mu_{p_j}(p_j) = \mu_{p_j}(f_j(\vec{d}))$:

$$\mu_o = \mathcal{P}(\mu_{d_1}, \dots, \mu_{d_n}, \mu_{p_1}, \dots, \mu_{p_q}).$$

The *combination function* \mathcal{P} reflects the design or trade-off strategy, which indicates how competing attributes of the design should be traded-off against each other [1].

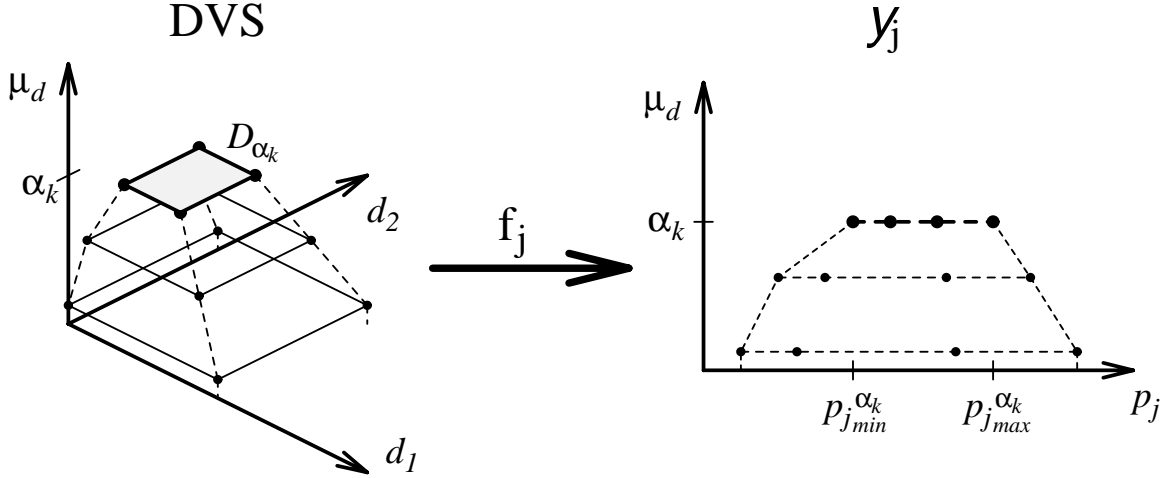


FIGURE 1. The Level Interval Algorithm.

Imprecision Calculations

After specifying design preferences $\mu_{d_1}, \dots, \mu_{d_n}$ and functional requirements $\mu_{p_1}, \dots, \mu_{p_q}$, and identifying the appropriate design strategy, the individual μ_{d_i} are combined to obtain μ_d , the combined design preference on the DVS. μ_d is then induced onto the PVS, using the extension principle [4]:

$$\mu_d(\vec{p}) = \sup\{\mu_d(\vec{d}) \mid \vec{p} = \vec{f}(\vec{d})\}$$

where sup over the null set is defined to be zero. $\mu_d(\vec{d})$ is the combined design preference on the DVS, as distinct from $\mu_d(\vec{p})$, the combined design preference induced onto the PVS. $\mu_d(\vec{p})$ is obtained by mapping $\mu_d(\vec{d})$ onto the PVS.

Previously, $\mu_d(\vec{p})$ has been calculated using the *Level Interval Algorithm*, or *LIA* [2], first proposed by Dong and Wong [5] as the “Fuzzy Weighted Average” algorithm and also called the “Vertex Method”. The LIA uses M design preference α -cuts $D_{\alpha_1}, \dots, D_{\alpha_M}$ in the DVS to calculate individual induced α -cut intervals $[p_{j_{\min}}^{\alpha_k}, p_{j_{\max}}^{\alpha_k}]$ which together define the induced α -cuts P_{α_k} in the PVS:

$$\begin{aligned} D_{\alpha_k} &= \{\vec{d} \in \text{DVS} \mid \mu_d(\vec{d}) \geq \alpha_k\} \\ &= [d_{1_{\min}}^{\alpha_k}, d_{1_{\max}}^{\alpha_k}] \times \dots \times [d_{n_{\min}}^{\alpha_k}, d_{n_{\max}}^{\alpha_k}] \\ P_{\alpha_k} &= \{\vec{p} \in \text{PVS} \mid \mu_d(\vec{p}) \geq \alpha_k\} \\ &= [p_{1_{\min}}^{\alpha_k}, p_{1_{\max}}^{\alpha_k}] \times \dots \times [p_{q_{\min}}^{\alpha_k}, p_{q_{\max}}^{\alpha_k}] \end{aligned}$$

where $k = 1, \dots, M$. For each α_k , the LIA evaluates $p_j = f_j(\vec{d})$ for the 2^n permutations of α -cut end points which correspond to the corners of an n -cube defined by D_{α_k} (there are n design variables and M α -cuts). Figure 1 illustrates how α -cuts D_{α_k} in two design variables d_1 and d_2 are induced onto the interval $[p_{j_{\min}}^{\alpha_k}, p_{j_{\max}}^{\alpha_k}]$. f_j is evaluated at the $2^n = 4$ corner points of each D_{α_k} rectangle. It is assumed that $p_{j_{\min}}^{\alpha_k}$ and $p_{j_{\max}}^{\alpha_k}$ will occur at these corner points, and not inside D_{α_k} . This is not true in general: the mapping $f_j : \text{DVS} \rightarrow \mathcal{Y}_j$ and the combination function \mathcal{P} must satisfy certain conditions for the LIA to be exact [6]. In practice, these conditions require that f_j be monotonic: a severe restriction.

$\mu_{d_i}(d_i)$	individual design preference on \mathcal{X}_i (designer), $i = 1, \dots, n$
$\mu_d(\vec{d})$	combined design preference on the DVS
$\mu_d(\vec{p})$	combined design preference induced onto the PVS
$\mu_{p_j}(p_j)$	individual functional requirement on \mathcal{Y}_j (customer), $j = 1, \dots, q$
$\mu_p(\vec{p})$	combined functional requirement on the PVS
$\mu_p(\vec{d})$	combined functional requirement mapped back onto the DVS
$\mu_o(\vec{d})$	overall preference on the DVS
$\mu_o(\vec{p})$	overall preference on the PVS
\mathcal{P}	combination function
D_{α_k}	α -cut in the DVS, within which $\mu_d(\vec{d}) \geq \alpha_k$ ($k = 1, \dots, M$)
P_{α_k}	α -cut in the PVS, within which $\mu_p(\vec{p}) \geq \alpha_k$ ($k = 1, \dots, M$)

Optimization

The limitations of the LIA stem from the assumption that the extreme values of f_j will occur at the corner points of the D_{α_k} n -cube. The algorithm may thus be improved by relaxing this assumption. The problem restated is to find:

$$\begin{aligned}
 p_{j_{\min}}^{\alpha_k} &= \min\{p_j = f_j(\vec{d}) \mid \vec{d} \in D_{\alpha_k}\} \\
 p_{j_{\max}}^{\alpha_k} &= \max\{p_j = f_j(\vec{d}) \mid \vec{d} \in D_{\alpha_k}\}
 \end{aligned}$$

Finding extrema within a subspace is a constrained optimization problem.

Optimization techniques are divided into two categories: traditional and stochastic. Traditional methods converge in relatively few function evaluations but are less robust, tending to become stuck in local minima. Stochastic methods such as genetic algorithms are more robust, but require a large number of function evaluations. Where function evaluations are relatively expensive, as is common in design, traditional optimization methods are preferred.

The algorithm utilized here is Powell's method, which begins as a one at a time search. After each iteration a heuristic determines whether to replace the direction of maximum decrease with the net direction moved during the last iteration. This allows minimization down valleys while avoiding linear dependence in the set of search directions [7].

An important feature for a practical computational tool is a means to trade-off the number of function evaluations against accuracy. Such an adjustment enables the designer to use the same program to obtain quick estimates as well as precise evaluations. This is implemented as a user-specified fractional precision that defines termination criteria for the optimization algorithm.

Suppose that it is necessary to incur the minimum number of function evaluations. A fractional precision of 1 would be specified, creating automatically satisfied termination criteria, and the optimization would proceed through exactly one iteration of a one at a time search using the maximum step size. The algorithm begins at one corner of the search space D_{α_k} , and checks corners in each of the n directions given by d_1, \dots, d_n , moving to the minimum each time. It expends $n + 1$ function evaluations to find each end point, and therefore $2n + 2$ per α -cut, as compared to 2^n per α -cut for the LIA. This is a substantial improvement, but the α -cut interval obtained by this method is still only correct if f_j is monotonic.

If, however, f_j is known to be monotonic, $2n + 2$ is not the minimum number of function evaluations. The first pass of the optimization algorithm identifies the direction for each

d_i in which f_j increases. Subsequent extrema can then be directly evaluated, without the need for searching. Hence where f_j is monotonic, $n + 2$ function evaluations are required for the first α -cut and 2 for each subsequent α -cut.

Design of Experiments

Statistical design of experiments seeks to derive information about a process using as few observations as possible. It has two aims: to separate the effects to be measured from random noise, and to model the process with regression equations. The function f_j can be treated as an unknown process, which a shrewd optimization method should examine with a few function evaluations before searching for extrema. Statistical design of experiments is an efficient method to conduct this preliminary examination. Note that if the process is deterministic, *e.g.*, a computer program, repeated function evaluations will always give the same answer: the output contains no noise. Therefore statistical significance tests to distinguish the signal are unnecessary. This paper discusses only the use of experiment design to model the function, though statistical significance tests are a valuable technique for processes subject to noise.

Because the procedure is to be encoded in a computer program, advanced experiment design techniques cannot be implemented. Hence only a linear regression model will be fitted to the function. Such a simple model will adequately approximate the effect of few, if any, design variables, but that is sufficient. The purpose is not to replace the entire function with a sophisticated regression model, but rather to identify design variables with near-linear effects. For each of these variables, the function can then be approximated by a linear equation, shrinking the search space by one dimension.

The function should therefore be modeled over the search space D_{α_k} . But which D_{α_k} ? α -cuts with higher α are subsets of α -cuts with lower α . Thus D_{α_1} , the α -cut with the lowest α , contains all the other α -cuts. Any regression equations found to be acceptable over D_{α_1} will be acceptable over all α -cuts. Hence only one set of experiments over D_{α_1} is required to explore the entire search space.

The Imprecise Design Tool will use a 2 level experiment design with a center point to serve as a curvature check. A full factorial design would evaluate the same 2^n corner points of D_{α_k} as the LIA, but since there are n main effects and 1 average to be determined, only $n+1$ evaluations are strictly necessary (excluding the center point). A fractional factorial design, which only evaluates a balanced subset of corner points, is more efficient. In reducing a full factorial design down to a fractional factorial design, some interactions are unavoidably confounded with other interactions, so that their effects cannot be distinguished. It is assumed that main interactions, due to a single variable, are more likely than two-way interactions, which are in turn more likely than three-way and higher order interactions. Since only main effects will be estimated, they must not be confounded with each other. This requires a resolution III (or higher) design. But it is desirable for main effects also not to be confounded with two-way interactions, and this requires a resolution IV design [8].

For $n = 8$, the smallest resolution IV design is a 2^{8-4} fractional factorial design requiring 16 observations. A resolution III design would require 12 observations. Resolution III designs approach the strictly necessary $n + 1$ function evaluations. Resolution IV designs require between $2n$ and $4n - 4$ function evaluations [9]. For $n = 8$, 16 function evaluations would be used: 9 evaluations are strictly necessary to estimate the 8 main effects and 1 average, and so there are 7 “redundant” evaluations. But these evaluations are not wasted: they allow main effects to be separated from two-way interactions, and they provide 7 extra

points to verify the accuracy of the linear regression model.

The number of function evaluations can also be traded-off against accuracy for the design of experiments. The linear regression equations obtained replace the function where the approximation is acceptable. The criteria for “acceptable”, which determine how accurately the function is modeled, are directly related to the user-specified fractional precision used by the optimization algorithm. Thus a single parameter trades-off computational effort against accuracy for both optimization and experiment design.

Relaxing the criteria for model acceptability minimizes the number of function evaluations. Yet there are essential conditions that still must be satisfied: if the sign of the gradient for a design variable d_i is in doubt or if the center point is an extremum, the linear regression equation for d_i must be rejected. If the function is benign, a maximum of $4n - 3$ evaluations (including the center point) will be incurred to obtain the regression equations and up to 2 evaluations will be required for the predicted α -cut end points. $4n - 1$ evaluations exceeds the $n + 1$ evaluations required for a one at a time search, but the advantages are fourfold:

1. Monotonicity is not assumed: up to $3n - 5$ “redundant” points test for monotonicity and linearity.
2. The center point tests for curvature.
3. The entire data set is used in estimating each effect, instead of two points.
4. An even distribution of corner points is sampled, instead of $n + 1$ adjacent corners.

Discussion

Figure 2 shows the role of optimization and experiment design in the Imprecise Design Tool. The information flow for one performance variable $p_j = f_j(\vec{d})$ is shown. α -cut intervals $[p_{j_{\min}}^{\alpha_k}, p_{j_{\max}}^{\alpha_k}]$ can be calculated separately for each \mathcal{Y}_j , and then combined in the uppermost preference calculation module. All Method of Imprecision calculations that explicitly involve preference occur at this level. Mapping α -cuts from the DVS to \mathcal{Y}_j requires optimization, which searches for the induced α -cut interval $[p_{j_{\min}}^{\alpha_k}, p_{j_{\max}}^{\alpha_k}]$ that corresponds to D_{α_k} . To do so, the experiment design module is called repeatedly to evaluate the function f_j at different designs \vec{d} . The first time the module is called, it conducts a fractional factorial experiment over D_{α_1} , the α -cut with lowest α , and constructs linear regression equations. For subsequent calls, regression equations replace the function f_j for any design variables that are adequately approximated. The fractional precision δ used by the optimization and experiment design modules trades-off the number of function evaluations against accuracy.

For the two design alternatives in the turbofan engine design example in [3], the Imprecise Design Tool using the LIA required 12 and 128 function evaluations. The alternatives had 4 and 5 α -cuts with some coincident end points, especially for the first alternative. Although there were nominally 8 design variables for both alternatives, the number of dimensions in the search space was 3 and 6. Using only optimization, and without taking advantage of monotonicity, the current version of the Imprecise Design Tool required 12 and 38 function evaluations to obtain the same results. As expected, optimization has a greater advantage for larger n .

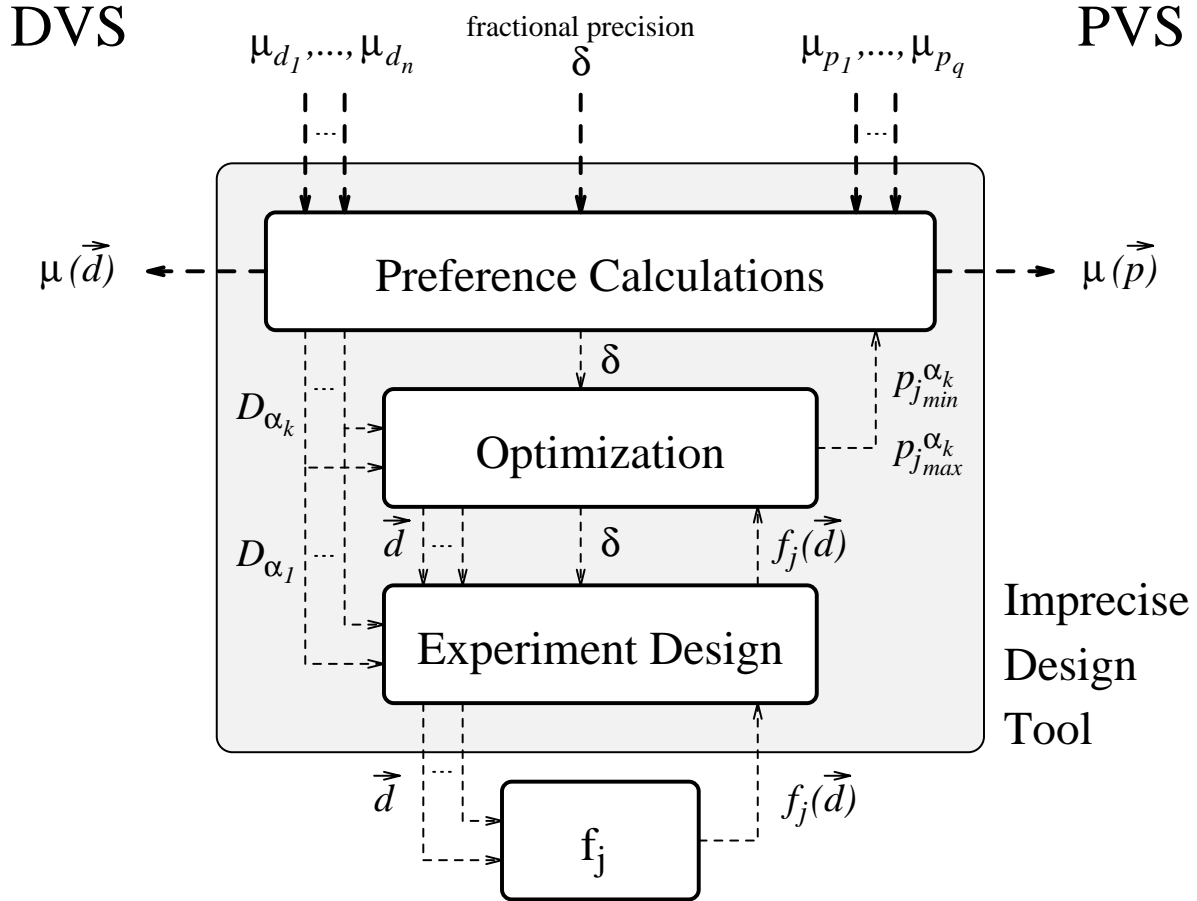


FIGURE 2. The Imprecise Design Tool.

Conclusion

This paper has presented new methods to perform Method of Imprecision calculations. These methods, which use optimization and experiment design techniques, provide two important enhancements:

1. Evaluation functions are no longer assumed to be monotonic.
2. The number of function evaluations required is substantially reduced.

Preliminary design necessarily deals with imprecise design descriptions. Traditional evaluation tools lack a systematic methodology to accommodate imprecision and deliver only disconnected information on individual designs. The Method of Imprecision uses the mathematics of fuzzy sets to represent and manipulate imprecise preliminary design information, enabling the designer to explore and understand the space of alternative designs in the context of the designer and customer's preferences among alternatives. This paper has presented continuing work in implementing this methodology in a practical computational design tool.

Acknowledgments

This material is based upon work supported, in part, by: The National Science Foundation under a NSF Grant No. DDM-9201424. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

References

- [1] Kevin N. Otto and Erik K. Antonsson. Trade-Off Strategies in Engineering Design. *Research in Engineering Design*, 3(2):87–104, 1991.
- [2] Kristin L. Wood, Kevin N. Otto, and Erik K. Antonsson. Engineering Design Calculations with Fuzzy Parameters. *Fuzzy Sets and Systems*, 52(1):1–20, November 1992.
- [3] William S. Law and Erik K. Antonsson. Including Imprecision in Engineering Design Calculations. In *Design Theory and Methodology – DTM '94*, volume DE-68, pages 109–114. ASME, September 1994.
- [4] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
- [5] W. M. Dong and F. S. Wong. Fuzzy weighted averages and implementation of the extension principle. *Fuzzy Sets and Systems*, 21(2):183–199, February 1987.
- [6] Kevin N. Otto, Andrew D. Lewis, and Erik K. Antonsson. Approximating α -cuts with the Vertex Method. *Fuzzy Sets and Systems*, 55(1):43–50, April 1993.
- [7] P. Adby and M. Dempster. *Introduction to Optimization Methods*. Chapman and Hall, London, 1974.
- [8] T. Barker. *Quality by Experimental Design*. Marcel Dekker, Inc., New York, 1985.
- [9] M. Phadke. *Quality Engineering Using Robust Design*. Prentice Hall, Englewood Cliffs, NJ, 1989.