AIAA JOURNAL
Vol. 38, No. 1, January 2000

From Dubious Construction of Objective Functions to the Application of Physical Programming

Achille Messac*
Northeastern University, Boston, Massachusetts 02115-5000

With the increasing availability of computational power, optimization is becoming a credible and viable option when designing complex multidisciplinary systems. Computational optimization generally involves three distinct phases: 1) model the physical system in terms of design parameters and design metrics, 2) form an aggregate objective function in terms of the design metrics, and 3) minimize the aggregate objective function using an optimization code. Robust analytical and computational tools are available to perform the first and third phases. The analytical tools available for constructing the objective function in phase two are remarkably simplistic and generally involve difficult-to-obtain weights. Because the optimum solution is only as effective as the aggregate objective function, any deficiency in the formation of the latter significantly impacts the ultimate outcome. The multiobjective design optimization process is examined from the perspective of constructing objective functions. We expose the shortcomings of weight-based methods using analytical and numerical means. Through analytical, graphical, and computational means, we show how the physical programming approach entirely circumvents the reliance on weight, thereby resulting in a new method of practical and general applicability.

I. Introduction

MULTIOBJECTIVE design optimization can be regarded as comprising three distinct phases: 1) model the system and develop design metrics in terms of design parameters, 2) quantify the overall design objective by forming an aggregate objective function, and 3) optimize the aggregate objective function using an optimization code. We examine in the following each of these phases.

The performance of phase one generally occurs within a mature computational and human infrastructure. Disciplinary expertise on the part of the engineer, together with the availability of robust commercial analysis codes, plays a pivotal role in the success of phase one. Consider for example the design of a system where step-response settling time (SRST) is one of the quantities of interest. The first step would be to model the system at hand using any number of analytical computational methods. Once the means are available to simulate the step-response-time history, the engineer deduces the corresponding settling time (or related design metric). In actual designs many more such design metrics must be developed. Two related comments follow. First, the system modeling and the development of the design metrics, although possibly involving some challenges for high-order systems, are generally well understood. Second, the availability of a complete set of relevant design metrics unfortunately offers no insight regarding the correct form of the aggregate objective function.

The second phase, where the aggregate objective function (AOF) is formed, is arguably the most critical. The AOF is expected to embody the full complexity of the wishes, biases, and prejudices of the designer or customer. Specifically, the AOF reflects the objective laws of nature (e.g., $F = ma$) as well as the subjective designer-imposed constraining environment (e.g., a noisy car is undesirable). The process of forming an AOF is as much an art as it is a science. Such methods as the weighted-sum and the weighted-square-sum are ineffective in expressing real-world preferences.[1–4] This paper presents an examination of the process of forming AOF and explains how physical programming[1,5–8] offers particular advantages.

Utility theory[9,10] offers one of the most comprehensive methods for formulating and modeling the decision-making process in a multiobjective environment. It helps the design researcher define a set of axiomatic properties for generic objective functions (utility functions). A strong practical limitation of utility theory is its lack of guidance as to how one should construct utility functions for real-world problems. Similar concerns are expressed in Refs. 11–14. To address this deficiency, we need a method for the systematic development of AOF that correctly reflects the full complexity of the designer's preference. Whereas utility theory defines in great theoretical detail the required properties of utility functions, it provides no practical means for the general development of utility functions. Physical programming exploits important aspects of utility theory to provide a methodical approach for developing preference functions (the analog of utility functions).

In their paper entitled "Optimization of Design Utility," Thurston et al.[13] express many issues that are directly related to the preceding discussion. They correctly point to the necessity to keep sight of what constitutes a good design throughout the design process. The single aggregate objective function that is minimized must judiciously represent the truly multiobjective decision-making process. They note that we evaluate the final design with respect to each objective and not in terms of the single aggregate objective (as the computational procedure would suggest). How then do we make sure that the single objective function is correct? This seemingly trivial question should play a critical role in computational engineering design.

In the third phase of multidisciplinary design optimization (MDO) application, a robust computational optimization code is employed to optimize the AOF subject to constraints. This phase is generally well understood, and the corresponding analytical and computational tools are mature. The various available tools are generally classified with respect to many considerations that might include continuous/discrete variables, integer/noninteger variables, convex/nonconvex problems, and linear/nonlinear constraints. For each such category, many robust commercial codes are available.

Since the early 1980s, interest in the area of engineering design in a multidisciplinary setting has grown.[15–21] The publications just cited focus on the area of control-structure integrated design (CSID), which is computationally intensive. Because of the computational intensity of the CSID problem, it is very costly to be optimizing with an incorrect objective function, though this unfortunately happens routinely. This situation was a strong motivation for the development of physical programming. Physical programming distinguishes itself by its ability to systematically develop an AOF that correctly reflects the full complexity of the decision-maker's wishes. It is appropriate to note that this paper's direct contribution is in the area of multiobjective optimization. However, because MDO is generally multiobjective in nature, this paper's findings are also relevant to MDO.

Received 26 August 1998; revision received 2 May 1999; accepted for publication 19 May 1999.

*Associate Professor, Department of Mechanical Engineering, Multidisciplinary Design Laboratory; messac@coe.neu.edu. Associate Fellow AIAA.

At this point it is important to note that, ultimately, designers are chiefly interested in having the means to uncover the most preferred design. In evaluating the effectiveness of an optimal design infrastructure, it may be appropriate to pose the following fundamental questions: 1) Given the appropriate set of design metrics, is it possible for the structure of the objective function being used to yield the most preferred solution, at all? 2) If no, then the objective function in question is inadequate. (Note that this unfortunate occurrence is common with regard to the popular weighted-sum objective function method.) If yes, then how easy is it to obtain this most preferred solution in practice? In particular, does the formation of the correct objective function require the designer to attempt to identify—in an ad hoc manner—the values of physically meaningless scalar weights? In the following, we present a pragmatic perspective of the related state of the art and of the new possibilities that the physical programming methodology brings to the fore.

This paper is organized as follows. The next section presents two simple examples that are used throughout the paper both for discussion purposes and for obtaining numerical results. Section III reviews some of the prevailing methods for forming AOF. A synopsis of the physical programming method is presented in Sec. IV. In Sec. V the examples (presented in Sec. II) are optimized using both the physical programming and the weighted-sum methods, leading to some interesting results and discussions. Section VI provides concluding remarks.

II. Discussion Examples

We define in this section two examples that we use throughout the paper both to help focus the discussion and later to provide numerical results. The first involves one design parameter and two design metrics, whereas the second involves two design parameters and three design metrics. We chose these simple examples to succinctly bring to the fore the salient points of this paper.

Example 1: The first example is a pinned-pinned beam of length L, width b, and height h (Fig. 1). A load of magnitude P is applied at the center of the beam. The volumetric mass density and Young's modulus are respectively denoted by ρ and E. The quantities of interest to the designer, or design metrics, are the midspan displacement and the beam mass, respectively, given by

$$\mu_1 = M = \rho bhL \tag{1}$$

$$\mu_2 = \delta = \frac{PL^3}{4Ebh^3} \tag{2}$$

The single design parameter is the beam height h, formally expressed as

$$x_1 = h \tag{3}$$

The designer's preferences are both qualitative and quantitative in nature. The qualitative preferences are to 1) minimize the midspan deflection and 2) minimize the beam mass. The quantitative preferences are discussed in Secs. IV and V within the physical programming context.

Example 2: The second example involves the same beam. The designer's preferences are now expressed in terms of three design metrics that depend on two design parameters. The design metrics are

$$\mu_1 = M = \rho bhL \tag{4}$$

$$\mu_2 = \delta = \frac{PL^3}{4Ebh^3} \tag{5}$$

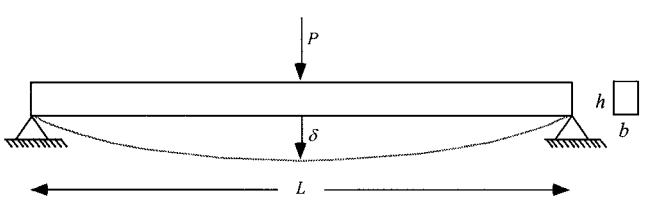

Fig. 1 Beam example.

$$\mu_3 = b \tag{6}$$

The two design parameters are

$$x_1 = h \tag{7}$$

$$x_2 = b \tag{8}$$

We observe that the quantity b is both a design metric and a design parameter. This point is discussed later.

III. Formation of Objective Functions

This section reviews some of the prevailing methods for forming AOF. This brief review will help explain the need for the unique features of physical programming. As a part of this discussion, we first address certain terminology-related issues that are often a source of confusion. For example, we seek to draw attention to the distinction between the terms design metric and objective function, as used in this paper.

A discussion of the terminology used in this paper, together with relevant editorial notes, follows. For a more extensive discussion of the general state of the engineering design lexicon, the interested reader may refer to Ref. 22, where it is shown that the objective of promoting intelligible discourse within the design community is entirely germane to the overriding goal of producing better designs.

A. Design Parameter

A design parameter is a parameter over which the designer has direct control. The variable x denotes the vector of design parameters. Other commonly used terms are design variables, decision variables, or decision parameters. An example of a design variable may be the length of a beam.

B. Design Metric

A design metric refers to an objective measure of a design attribute. The variable $\mu(x)$ denotes the vector of design metrics, which is generally expressed in terms of the design parameter vector. A design metric vector may comprise such quantities/measures as the mass, the price, the area moment of inertia, or the fundamental frequency of a beam.

We make the important note that the term design metric, as used in this paper, does not reflect the subjective preferences of the designer/decision maker. Expressing the designer's preference as a mathematical function is a complex task. For example, the designer might want to minimize the mass, while maximizing the fundamental frequency; maximizing the fundamental frequency beyond a certain point might not be as important as the objective of minimizing cost. These observations partially illustrate the difficulty of mathematically capturing the full complexity of the designer's wishes.

We also note that a given quantity, e.g., the length of a beam, may be both a design parameter (because the designer has control over its value) and a design metric (because it will also be used to evaluate the goodness of the beam design). The designer might prefer a short beam, for example. Typical problem formulations usually ascribe a very restricted set of desired behaviors to design parameters (e.g., equal to a constant, less than a constant, or between two constants). We show in Sec. V how physical programming facilitates the expression of preference regarding design parameters as well as design metrics.

Other terms used in the literature as synonyms of design metric include design attribute, design criterion, objective function, performance function, performance index, performance parameter, merit function, figure-of-merit, goal, and performance metric. This liberal lexicon is often the source of much confusion because it fails to distinguish clearly between the objective design metric and the subjective preference function, as defined in the sequel.

To conclude this discussion of terminology, we note that this paper is not concerned with the means for developing design metrics (such a discussion would be domain-dependent), but rather with the means for employing design metrics for constructing aggregate objective function.

C. Preference Function

A preference function (usually referred to as objective function), an explicit function of a single design metric, reflects the subjective preference of the designer with regard to that design metric. The variable $P(\mu)$ denotes the vector of preference functions. As an example, if the designer wishes to maximize the fundamental frequency of the beam f (using a minimization code), the corresponding preference function might take the form $P_i(f) = \alpha/f$, where α is a constant that implicitly reflects the importance of maximizing f relative to other objectives (e.g., minimizing mass). In the case of maximization, any strictly decreasing monotonic function would be an appropriate candidate, but the inverse function just used is unduly inflexible. In addition, the use of only one constant (α) is inadequate, as it offers minimal flexibility. We note that the term objective function is generally meant to remind us that the objective of the designer is reflected in the said function. As such, the preference function reflects a subjective statement, which is chiefly distinct from a design metric and which reflects an objective statement (e.g., the mass of a beam). The recognition of this distinction plays a critical role in the proper development of an appropriate preference function.

All of the terms just used as synonyms of design metric are also used as synonyms of preference function in the literature, thereby making it impossible to clearly distinguish between preference function and design metric.

D. Design Constraint

A design constraint is a statement of the form $l_i \leq (\mu_i \text{ or } x_i) \leq \mu_i$, where l_i and μ_i, respectively, denote lower and upper bounds (which need not be finite).

E. Aggregate Preference Function

An aggregate preference function (APF) J is a scalar function of the preference function vector that, together with the design constraints, completely represents the designer preferences (at least as stated). The term APF is not used in the literature. Instead, a large number of terms are used, the most common being objective function. The term objective function is unfortunately often used to mean design metric, preference function, as well as AOF. The term objective space usually implies the existence of a multidimensional space spanned by a set of objective functions. Using the terminology just defined, we can express the distinction between design metric space, preference function space, and APF space.

Once the APF is available, numerical or analytical optimization may proceed. It is the aggregation (or scalarization) process of the set of preference functions that this paper intends to examine. Next, we discuss the prevailing methods for forming APF.

F. Aggregate Preference Functions: Examples

We review in the following some of the main methods available for forming the aggregate preference functions. The quantity w_i denotes a generic constant scalar weight, and G_i and B_i respectively denote generic good and bad values pertaining to the ith design metric.

Absolute value:

$$J_{\text{av}} = \sum_i w_i |\mu_i(x) - G_i| \tag{9}$$

or, equivalently,

$$J_{\text{av}} = \sum_i \left| \frac{\mu_i(x) - G_i}{B_i - G_i} \right| \tag{10}$$

The absolute value option is unfortunately not easily implemented in most analytical settings. One of the reasons for this difficulty is that it is not smooth.

Weighted-sum:

$$J_w = \sum_i w_i \mu_i(x) \tag{11}$$

To construct the APF, the designer has at his/her disposal only a set of constants that merely change the slopes of a hyperplane in the design-metric space. Marginal utility is unfortunately/incorrectly modeled as being always constant in the whole design-metric space.[9]

Weighted-square-sum:

$$J_w = \sum_i w_i [\mu_i(x) - G_i]^2 \tag{12}$$

In this case two constants per design metric are available to tailor the shape of the APF.

Weighted-maximum:

$$J_{\max} = \max_i \frac{\mu_i(x) - G_i}{B_i - G_i} \tag{13}$$

This form, in practice, presents smoothness-related difficulties.

Substitute objective function[19]:

$$J_{\text{sub}} = \prod_i \frac{\mu_{i,\max} - \mu_i(x)}{\mu_{i,\max} - \mu_{i,\min}} \tag{14}$$

where $\mu_{i,\min}$ and $\mu_{i,\max}$ respectively denote the minimum and maximum values of μ_i, when μ_i is optimized while all other metrics are free.

Kreisselmeir–Steinhauser function[17]:

$$J_{\text{KS}} = \frac{1}{\rho} \ln \sum_i \exp[\rho \mu_i(x) - G_i] \tag{15}$$

where ρ is a parameter that is increased as the optimum is approached during optimization.

Distance from the utopia point:

$$J_w = \sum_i w_i [\mu_i(x) - \mu_{i,\min/\max}]^2 \tag{16}$$

where it is clearly understood that the APF will, in general, not vanish.

The common thread in the preceding options is the existence of certain parameters that are intended to shape the hypersurface of the aggregate preference function—in design-metric space. Importantly, these parameters are usually void of any physical significance. This observation stands in contrast to the physical programming paradigm.

IV. Physical Programming Synopsis

Physical programming is a new approach to computational design optimization.[1] The application of physical programming is intended to enhance an engineer's ability to obtain an optimal design by employing a flexible and more natural problem formulation framework. Under the physical programming paradigm, the designer does not need to specify optimization weights in the problem formulation phase. Rather, the designer specifies ranges of different degrees of desirability for each design measure.

Consider the design metrics of example 1: the beam mass M and the midspan deflection δ. Design specifications may require $M \leq 350$ kg and $\delta \leq 0.15$ m, and a designer may also wish to minimize each such metric without violating these hard specifications. In this particular case the designer may create an aggregate preference function of the form

$$J_w = w_1 M + w_2 \delta \tag{17}$$

and manipulate the weights w_1 and w_2 until the minimized AOF leads to acceptable values for M and δ. The designer will in general need to inject considerable bias or subjectivity into the problem formulation. He or she may accept the values of 351 for M and 0.12 for δ, but might much prefer 345 for M and the higher value of 0.135 for δ. In the parlance of multiobjective optimization, the second option is not dominated by the first. However, the designer cannot explicitly articulate this wish using the traditional AOF just shown. This inability to effectively articulate preference—by having to rely on weights—is a key limitation of conventional optimization.

An alternative to the preceding scenario is to monitor the optimization process and to change the weights during the solution process periodically, as opposed to performing repeated full optimization runs. However, no general and robust method exists to perform this task. Moreover, when a large number of conflicting metrics exists, the task of tweaking the weights for the different design metrics becomes prohibitively unwieldy.

By allowing the designer to express preferences flexibly and concisely, physical programming addresses the just-mentioned limitation of traditional optimization. Simultaneously, physical programming obviates the need to adjust weights. We first describe how a designer might express a realistic design preference. The preference ranges for the beam mass could be stated as follows.

1) Highly desirable: <250 (kg)
2) Desirable: 250–275
3) Tolerable: 275–300
4) Undesirable: 300–325
5) Highly undesirable: 325–350
6) Unacceptable: >350

Note that this description encompasses both hard and soft preferences. The latter are expressed in terms of ranges of differing degrees of desirability and the former in terms of inequalities or equalities.

In addition to eliminating the need to specify and adjust weights, physical programming is also ideally suited to address the inherent multiobjective nature of design problems, where multiple conflicting objectives govern the search for the best solution. To reconcile the objectives and find a compromise solution, the designer usually engages in an iterative weight-tweaking process that does not necessarily converge. The convergence property of this human-in-the-loop process depends as much on the numerics of the problem as it does on the level of expertise of the designer in the art and science of optimization.

Physical programming provides a more deterministic approach to obtaining a solution that satisfies the designer's preferences. A detailed development of the physical programming approach is provided in Ref. 1. References 5–8 present further physical programming work. A synopsis of the physical programming implementation procedure is provided in the sequel.

A. Physical Programming Application Procedure

This section describes the procedure for applying physical programming, assuming an operational software implementation thereof is available. Alternatively, the theoretical development of physical programming presented in Ref. 1 can be coded by the user. The physical programming implementation that is used for this paper is embodied in the software package entitled PhysPro, which is MATLAB®-based[23] and which is at a more advanced stage than that presented in Ref. 1. We define in the sequel the steps that must be followed in the application of physical programming.

Classification of Preferences (Preference Functions)

Within the physical programming procedure, an engineer expresses preferences with respect to each design metric using four different classes. Each design belongs to one of four generic classes. Figure 2 depicts the qualitative meaning of each class. A generic design metric μ_i is on the horizontal axis, and the function that will be minimized for that metric P_i, hereby called the preference function, is on the vertical axis. Each class comprises two subclasses, hard and soft, referring to the sharpness of the preference. All soft class functions will become constituent components of the APF.

The desired behavior of a generic measure is described by one of eight subclasses, four soft and four hard. These subclasses are illustrated in Fig. 2 and are characterized by the following.

1) Soft:
   a) Class-1S: Smaller-is-better (minimization)
   b) Class-2S: Larger-is-better (maximization)
   c) Class-3S: Value-is-better
   d) Class-4S: Range-is-better
2) Hard:
   a) Class-1H: Must be smaller; i.e., $\mu_i \leq \mu_{i,\max}$
   b) Class-2H: Must be larger; i.e., $\mu_i \geq \mu_{i,\min}$

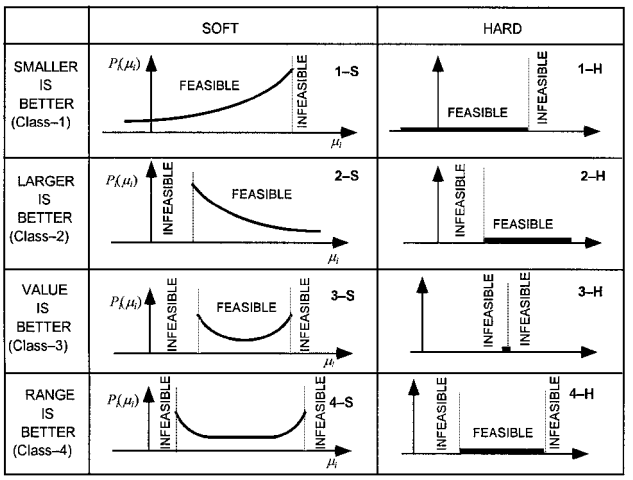

Fig. 2 Classification of design metrics.

   c) Class-3H: Must be equal; i.e., $\mu_i = \mu_{i,\text{val}}$
   d) Class-4H: Must be in range; i.e., $\mu_{i,\min} \leq \mu_i \leq \mu_{i,\max}$

where $\mu_{i,\max}$, $\mu_{i,\min}$, and $\mu_{i,\text{val}}$ represent given constants.

Within conventional design optimization approaches, the design metrics for which class 1S or 2S apply would generally become part of the AOF with a multiplicative weight, and all of the hard classes would become constraints. Cases of class 3S and 4S would be significantly more difficult to treat: one approach would be to use a positive or negative weight, depending on the current value of the corresponding measure during optimization. A large amount of trial and error would be involved in choosing the right weights. Physical programming removes this trial and error entirely by using the carefully structured preference functions.

The preference functions shown in Fig. 3 provide the means to express a set of ranges of differing desirability for a given design metric. As shown in Fig. 3, the soft preference functions provide information that is deliberately imprecise. By design, the infimum of any preference function is zero. Next, we explain how quantitative specifications are associated with each design metric. (Note that class 4S actually represents an extension of the work in Ref. 1.)

Physical Programming Lexicon

Physical programming allows an engineer to express preferences with regard to each design metric with more specificity than by simply saying minimize, maximize, greater than, less than, or equal to. Physical programming explicitly recognizes the limitations of such a problem formulation framework, and addresses these by employing a more descriptive lexicon. This lexicon comprises terms that characterize the degree of desirability of 6 ranges for each generic soft design metric for classes 1S and 2S, 10 ranges for class 3S, and 11 ranges for class 4S. To illustrate, consider the case of class 1S, shown in Fig. 3. The ranges are defined as follows in order of decreasing preference.

1) Highly desirable range ($\mu_i \leq v_{i1}$): An acceptable range over which the improvement that results from further reduction of the performance measure is desired, but is of minimal additional value.
2) Desirable range ($v_{i1} \leq \mu_i \leq v_{i2}$): An acceptable range that is desirable.
3) Tolerable range ($v_{i2} \leq \mu_i \leq v_{i3}$): This is an acceptable, tolerable range.
4) Undesirable range ($v_{i3} \leq \mu_i \leq v_{i4}$): A range that, while acceptable, is undesirable.
5) Highly undesirable range ($v_{i4} \leq \mu_i \leq v_{i5}$): A range that, while still acceptable, is highly undesirable.
6) Unacceptable range ($\mu_i \geq v_{i5}$): The range of values that the generic measure must not take.

The parameters v_{i1}–v_{i5} are physically meaningful constants that are specified by the designer to quantify the performance objectives, or preferences, associated with the ith design metric. These parameters delineate the ranges for each design metric. Reference 1 discusses the quantitative implications of the preceding definitions

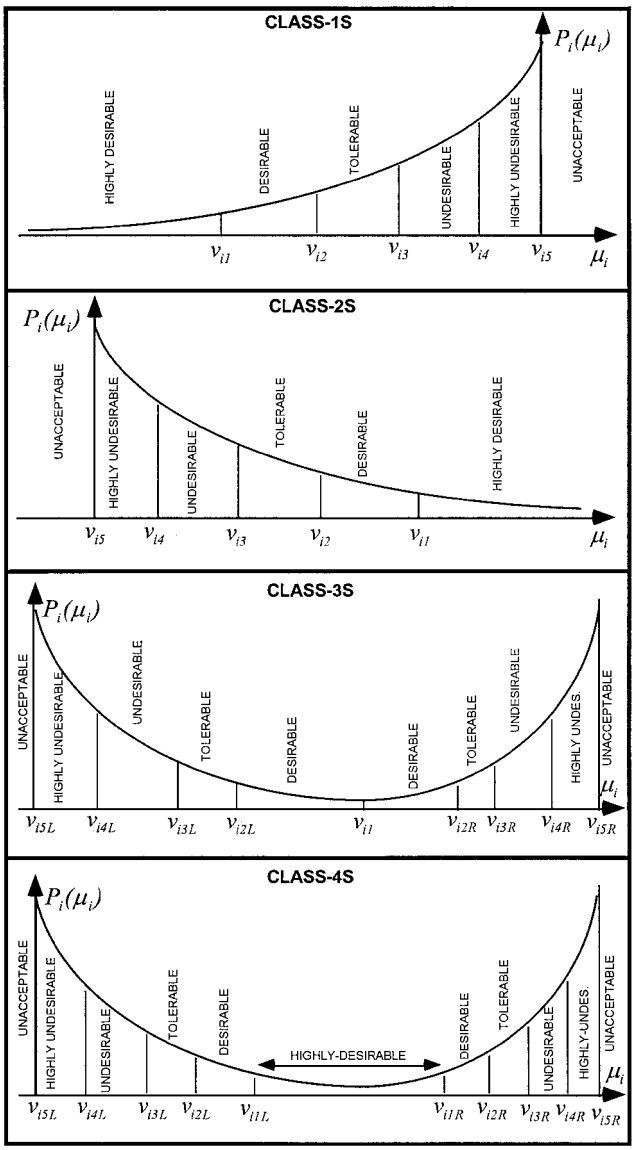

Fig. 3 Preference functions for design metrics.

and provides the mathematical procedure for developing the preference function for each generic design metric.

The preference functions map design metrics, such as M, into nondimensional, strictly positive real numbers. This mapping, in effect, transforms disparate design metrics with different physical meanings onto a dimensionless scale through a unimodal convex function. Figure 3 loosely illustrates the mathematical nature of the preference functions and shows how these functions allow the designer to express the regions of differing degrees of goodness for a given design metric. Consider the first curve of Fig. 3: the class function for class 1S design metrics. Six ranges are defined. The parameters v_{i1}–v_{i5} are specified by the designer. When the value of the design metric μ_i is less than v_{i1} (highly desirable range), the value of the preference function is small, which requires little further minimization of the preference function. When, on the other hand, the value of the design metric μ_i is between v_{i4} and v_{i5} (highly undesirable range), the value of the class function is large, which requires significant minimization of the class function. The behavior of the other preference functions is indicated in the figure. Preferences regarding each design metric are treated independently, allowing the inherent multiobjective nature of the problem to be preserved.

This discussion tersely presented the basic idea of physical programming. The value of each generic preference function (for each design metric) governs the optimization path in design metric space,

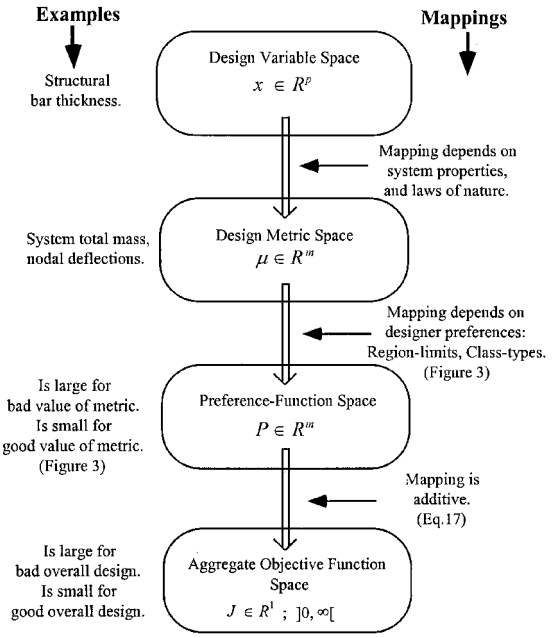

Fig. 4 Physical programming mappings.

and the preference function is constructed deterministically and involves no physically meaningless weight iteration.

B. Physical Programming Mappings

We now briefly discuss the various mappings that take place in the implementation of physical programming. These mappings will define the path from design variables to the aggregate preference function, which is the actual function that the nonlinear programming code minimizes. Figure 4 shows these various mappings. As illustrated in Fig. 4, we begin with the design variables x. The design variables are mapped into the preference function space μ_i using the Metrics Evaluation Module (MEM) created by the designer for a given problem. The value of a given preference function P_i depends 1) on the value of the corresponding design metric, 2) on the class type assigned to that design metric (e.g., smaller-is-better), and 3) on the values (v_{i1}–v_{i5}) assigned to the desirability ranges. Loosely speaking, the sum of all of the preference functions equals the APF (see Ref. 1 for details). We note, however, that the preference functions are not independently formed; there is an assumed intermetric preference.

C. Physical Programming Application Steps

With this background in physical programming terminology and mappings, we are prepared to explain the procedure for solving a problem using physical programming.

1) Step 1: Create a software module (MEM) that uses the current numerical values of the design parameters $\boldsymbol{x}$ as input and that provides the corresponding numerical values for the design metric vector μ_i as output (i.e., a software module that models the behavior of the physical system).

2) Step 2: Specify the class type for each design metric (1S-4H) (see Fig. 2).

3) Step 3: Provide the range limits for each design metric (see Fig. 3). The designer specifies 5 limits for classes 1S or 2S, 9 for 3S, and 10 for 4S. For the hard classes the designer specifies one limit for classes 1H, 2H, or 3H, and two limits for 4H (see Fig. 2). The set of these range limits, when recorded in tabular form, constitutes the preference table.

4) Step 4: Solve the constrained minimization that is defined by Eqs. (17–26). Note that, if a physical programming software implementation is available to the designer, steps 1–3 constitute the extent of the designer's involvement with nonlinear programming. Rather than engaging in tweaking weights, the designer can instead explore the consequences of the various physically meaningful preference choices.

Table 1 Preference table

		HU[a]	U[b]	T[c]	D[d]	HD[e]	D	T	U	HU
Variables	Class	v_{i5}	v_{i4}	v_{i3}	v_{i2}	v_{i1}	v_{i1}	v_{i2}	v_{i3}	v_{i4}	v_{i5}
					Example 1						
M	1S	——	——	——	——	——	250	275	300	325	350
δ	1S	——	——	——	——	——	0.01	0.05	0.10	0.125	0.150
					Example 2						
M	1S	——	——	——	——	——	250	275	300	325	350
δ	1S	——	——	——	——	——	0.01	0.05	0.10	0.125	0.15
b	2S	0.02	0.025	0.030	0.050	0.100	——	——	——	——	——

[a]HU, highly undesirable. [b]U, undesirable. [c]T, Tolerable. [d]D, desirable. [e]HD, highly desirable.

D. Physical Programming Problem Model

The physical programming problem model takes the form

$$\min_x P(\mu) = \frac{1}{n_{sc}} \sum_{i=1}^{n_{sc}} P_i[\mu_i(x)] \qquad \text{(for soft classes)} \qquad (18)$$

subject to

$$\mu_i(x) \le v_{i5} \qquad \text{(for class 1S metrics)} \qquad (19)$$

$$\mu_i(x) \ge v_{i5} \qquad \text{(for class 2S metrics)} \qquad (20)$$

$$v_{i5L} \le \mu_i(x) \le v_{i5R} \qquad \text{(for class 3S metrics)} \qquad (21)$$

$$v_{i5L} \le \mu_i(x) \le v_{i5R} \qquad \text{(for class 4S metrics)} \qquad (22)$$

$$\mu_i(x) \le v_{i,\max} \qquad \text{(for class 1H metrics)} \qquad (23)$$

$$\mu_i(x) \ge v_{i,\min} \qquad \text{(for class 2H metrics)} \qquad (24)$$

$$\mu_i(x) = v_{i,\text{val}} \qquad \text{(for class 3H metrics)} \qquad (25)$$

$$v_{i,\min} \le \mu_i(x) \le v_{i,\max} \qquad \text{(for class 4H metrics)} \qquad (26)$$

$$x_{j,\min} \le x_j \le x_{j,\max} \qquad \text{(for design parameter constraints)} \qquad (27)$$

where $v_{i,\min}$, $v_{i,\max}$, $v_{i,\text{val}}$, $x_{i,\min}$, and $x_{i,\max}$ represent prescribed values. The range limits are provided by the designer (see Table 1 and Fig. 3), and n_{sc} is the number of soft metrics that the problem comprises. Note that the aggregate preference function only comprises class functions associated with soft metrics. The hard metrics are treated as constraints.

The preceding problem model conforms to the framework of most nonlinear programming codes, with possible minor rearrangements. However, note that the form of the APF departs markedly from conventional design optimization methods. We now apply the preceding development to the two examples already described.

V. Example

A. Numerical Results

In this section we provide numerical results for the two beam examples already defined. Optimization results are given for each case using physical programming. We then present results involving other methods, which will allow us to make important comparisons.

Example 1: In example 1 we used the numerical constant values: $E = 10^{11}$ Pa, $\rho = 10^4$ kg/m^3, $P = 400$ N, $L = 10$ m, and $b = 0.05$ m. The design parameter h is also expressed in meters. In the parlance of physical programming, the designer's subjective preferences are expressed in Table 1, where the numerical values that fit the ranges of differing desirability are provided (the preference table). Note that Table 1 provides the v_{ij} preference values, as defined in Eqs. (19–22), which delineate the preference ranges. For example, the first row of numbers in Table 1 identically represents the mass-related preferences discussed in Sec. IV. No constraints are placed on the design parameter h. The optimization results are reported in Table 2 and Fig. 5.

To first contrast the weighted-sum and the physical programming approaches, the equivalent formulation of the optimization problem using the weighted-sum approach is obtained. (Reference 1 explains the relationship between equivalent weights and class functions.) In short, the weights are obtained using the derivative of the preference functions with respect to the design metrics, evaluated at the optimum solution. The weighted-sum problem statement reads

$$\min_h J_w = w_1 M(h) + w_2 \delta(h) \qquad (28)$$

The weights obtained are $w_1 = 0.228305$ and $w_2 = 230.538$. Using these weights in the problem formulation just stated, in Eq. (28) we find the same results as those obtained using physical programming. [The MATLAB® Optimization toolbox function constr[23] was used to solve Eq. (27).]

Table 2 Design parameters, design metrics, and objective function

	Example 1		Example 2	
Variables	Initial	Optimal	Initial	Optimal
Design metrics				
M	2,500	295	21	270.4
δ	16e-5	0.097	14.47	0.074
b	NA	NA	0.01	0.0331
Design parameters				
h	0.5	0.059	0.02	0.0778
b	NA	NA	0.01	0.0331
Objective function J	17,331	6.76	388,108	5.006

Example 2: In example 2 we used the same numerical constant values as those used in example 1, with two exceptions: $L = 10.5$ m and b is now a free design parameter. The designer's preferences are expressed in the preference table (Table 1). Again, no constraints are placed on the design parameters h and b. The physical programming optimization results are reported in Table 2 and shown in Fig. 5.

As in the case of example 1, we determined the equivalent formulation for the weighted-sum approach. The weighted-sum problem statement is found in the form

$$\min_{h,b} J_w = M + 1213.59708\delta - 5446b \qquad (29)$$

The preceding problem formulation [objective function, Eq. (29)] unfortunately yields an unstable stationary point, in particular, a saddle point (see Fig. 6, W4). In essence, the weighted-sum approach fails to be of any practical value in this case. We parenthetically note that this deficiency of the weighted-sum approach is rooted in its linear dependence on the design metrics, more specifically by its inability to capture solutions that lie on concave boundaries of the feasible design-metric space.[2–4]

Before we proceed with the discussion of the results, we provide objective functions—from other methods—that yield the same solution as physical programming. These objective functions take the following form.

Weighted-square sum:

$$J_w = 0.602349(M - 250)^2 + 231{,}960(\delta - 0.01)^2 + 1{,}000{,}000(b - 0.1)^2 \qquad (30)$$

Kreisselmeir–Steinhauser function:

$$J_{KS} = \ln[\exp(M - 277.43098) + \exp(\delta - 0) + \exp(1.60478 - b)] \qquad (31)$$

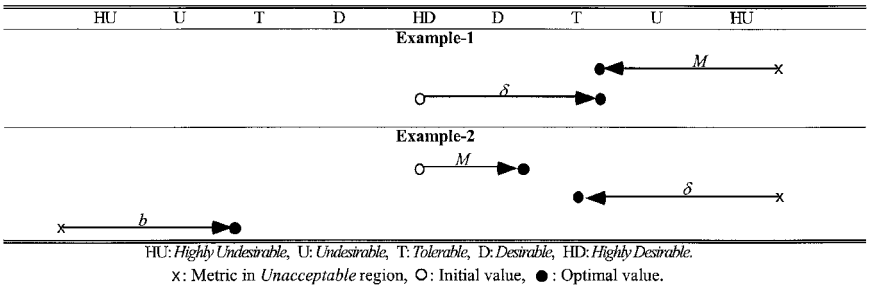

Fig. 5 Physical programming optimization results.

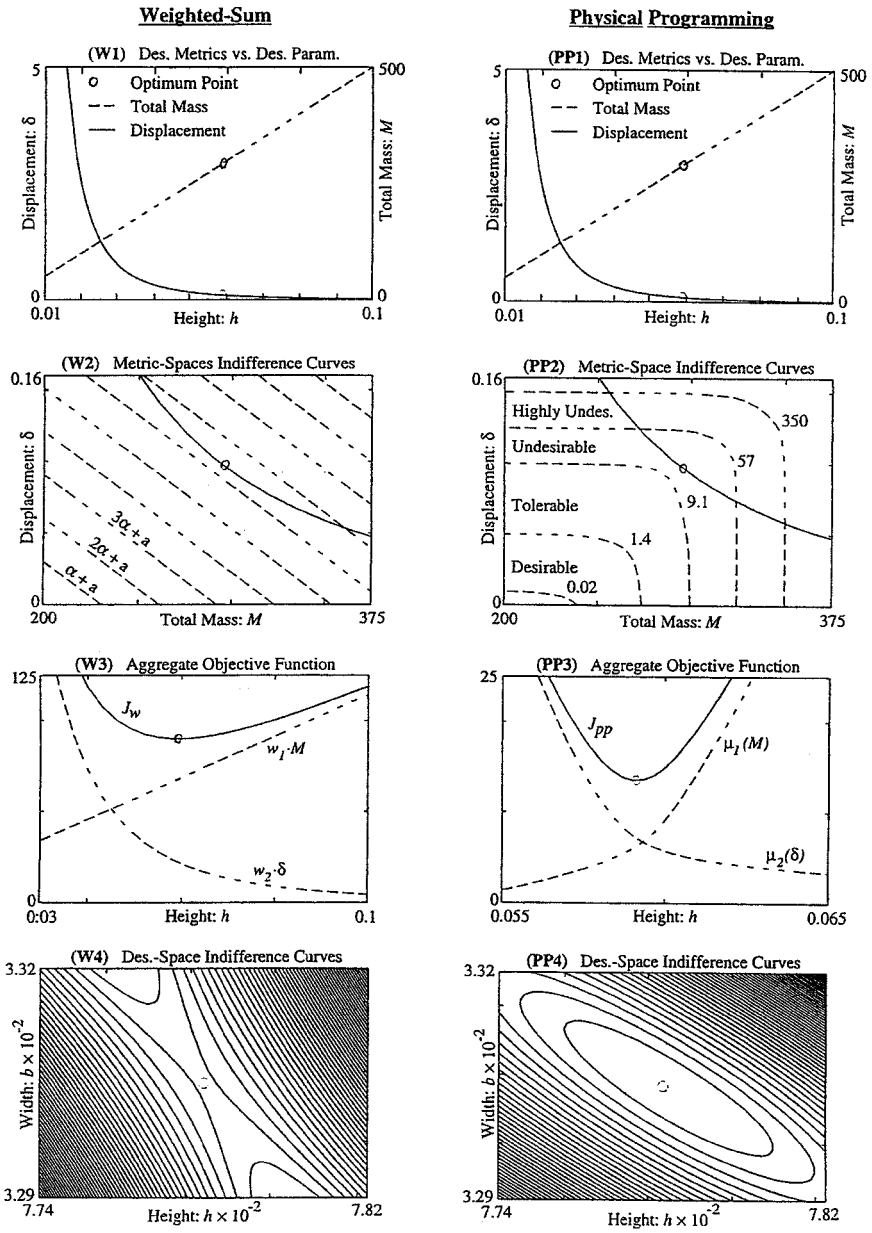

Fig. 6 Numerical results for examples 1 and 2.

Exponential weighted criteria[3]:

$$J_{\text{ewc}} = \exp(M/100) + 0.0010789\exp(100\delta) + 222.79\exp(-100b) \quad (32)$$

Weighted compromise programming[3]:

$$J_{\text{wcp}} = (0.0003947M)^3 + (\delta)^3 - (2.816758b)^3 \quad (33)$$

B. Results Discussion

Example 1: For example 1, Fig. 5 depicts the tradeoffs that took place during optimization. The beam mass and the midspan displacement (MSD) changed significantly. The mass improved by moving from the unacceptable region to the tolerable region, whereas the MSD worsened by moving from the highly desirable region to the tolerable region. The compromise solution is reached by prescribing physically meaningful values v_{ik}, rather than physically meaningless weights.

Figure 6 presents other aspects of the optimization process. The left and right columns respectively pertain to the weighted-sum and the physical programming approaches. The first three rows show results for example 1 and the last for example 2.

Panels (W1) and (PP1) show the plots of the design metrics vs the design parameter. As expected, there is no difference between the two approaches. We note that the two metrics are in conflict: M seeks to increase h, whereas δ seeks the reverse.

Panels (W2) and (PP2) show indifference curves (curves of constant preference) for both approaches. In the case of (W2), we see that the curves are straight lines that are equidistant and parallel, reflecting the fact that the preference surface is a plane. The weights in the weighted-sum approach are used to affect the slope of this plane. We note that the slope of the preference surface is the equivalent weight at the point of evaluation.[1] In the case of (PP2), the indifference curves are versatile and may take on complex shapes whose convexity is guaranteed. The shape of these curves is guided by the designer's choices in the form of region limits that are physically meaningful. Further, the gradient of the preference surface is allowed to change dramatically in preference space. This gradient change can be seen in panel (PP2), where the slopes are significantly smaller in the highly desirable region than they are in the highly undesirable region. (Observe the numerical values for the various indifference curves.) This change in gradient is essentially equivalent to changing the weights while optimizing, in such a way that the aggregate preference function correctly reflects the designer's preferences. The solid curve in panels (W2) and (PP2) represents a parametric function of the design metrics in terms of h (i.e., values of the design metrics as h varies). As expected, the solid curve is tangent to the optimal indifference curve.

Panels (W3) and (PP3) depict the APF in terms of the design parameter h. As expected, the optimum solution is the same in each case. However, the constituent individual preference functions (for different design metrics) are fundamentally different. In the case of the weighted-sum [Eq. (28)], the shape of each design metric is identical (within a multiplicative constant) to its pertaining preference function. In the case of physical programming, the shape of each individual preference function changes significantly (according to the designer's physical preferences) from that of the pertaining design metric. The plot pertaining to the mass is a straight line in both panels (W1) and (W3), but the contribution of the mass is no longer a straight line in the physical programming case [panels (PP1) and (PP3)].

Example 2: For example 2, Fig. 5 depicts the tradeoff that took place during optimization among the three design metrics. We recall that b is both a design variable and a design metric. As we see, letting b play a role in seeking a compromise solution presents no difficulty. Typically, we tend to limit our options in this regard: we let b be the subject only of inequality constraints because it is the design parameter.

In Fig. 6, panels (W4) and (PP4) convey an interesting situation. We first determine the equivalent weights.[1] Using these weights, which guarantee that the weighted-sum and the physical programming formulations will have a common extremum, we optimize the beam. The weighted sum case failed in this simple example. It fails in the following sense. If a designer does wish to obtain the solution provided by physical programming, the weighted sum will fail to satisfy this designer's wish. The weighted sum is structurally incapable of providing the sought-after solution in practice. The indifference curves for each approach are shown in these subplots. In the case of physical programming, we see a well-behaved minimum. In the weighted sum case the extremum is a saddle point, which means that in practice the correct solution cannot be obtained using the weighted-sum approach, even in this simple case. We are now ready to discuss the situation from a broader perspective.

Performance of other objective functions: Having established that the oft-used weighted-sum method failed in this simple example, it is appropriate to ask: How effective are other methods? The comprehensive answer to this question is the subject of ongoing investigation by these authors and others. In this paper we provide important observations that help to further characterize the physical programming method.

In Eqs. (30–33), we provided diverse objective functions that yield the same result as physical programming. The important question at this point is as follows: How easy is it to uncover the numerical values (weights) in these objective functions that will yield desirable results? Unfortunately, the answer to this question is not encouraging. For realistic problems, where the design metrics are nonlinearly coupled, determining correct meaningless weights is difficult. Even in the case of Eq. (31), where the numerical values do have physical interpretations, identifying these values is in general a complex task. The inherent structures of the preceding AOF are not flexible. By contrast, physical programming provides extensive flexibility to tailor the shape of the preference hyper-surface and does so through the designation of physically meaningful information (see the preference table). It is also interesting to note that, even in the preceding simple case, some of the preceding objective functions present appreciable numerical-conditioning difficulties. [To discover the numerical values in Eqs. (30–33), we used algebraic means with a priori knowledge of the final solution. Under normal circumstances, of course, this would not have been possible.]

VI. Concluding Remarks

This paper presented a discussion of one of the critical phases of computational optimization, namely that of constructing the aggregate objective/preference function. This paper studied the particular issues posed by the application of the weighted-sum and other methods and offered the physical programming method as an alternative that presents important practical benefits.

Acknowledgments

The author gratefully acknowledges support of this work by NSF through Grant DMI-9622652 and the CAREER Grant DMI-9702248. The assistance of Xuan Chen in helping to prepare some numerical data is hereby acknowledged.

References

[1]Messac, A., "Physical Programming: Effective Optimization for Design," *AIAA Journal*, Vol. 34, No. 1, 1996, pp. 149–158.

[2]Koski, J., "Defectiveness of Weighting Method in Multicriterion Optimization of Structures," *Communications in Applied Numerical Methods*, Vol. 1, No. 6, 1985, pp. 333–337.

[3]Athan, W. T., and Papalambros, P. Y., "A Note on Weighted Criteria Methods for Compromise Solutions in Multi-Objective Optimization," *Engineering Optimization*, Vol. 27, No. 2, 1996, pp. 155–176.

[4]Dennis, J. E., and Das, I., "A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimizations Problems," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 63–69.

[5]Messac, A., Gupta, S., and Akbulut, B., "Linear Physical Programming: Effective Optimization for Complex Linear Systems," *Transactions on Operational Research*, Vol. 8, Oct. 1996, pp. 39–59.

[6]Messac, A., and Hattis, P., "Physical Programming Design Optimization for High Speed Civil Transport (HSCT)," *Journal of Aircraft*, Vol. 33, No. 2, 1996, pp. 446–449.

[7]Messac, A., and Wilson, B., "Physical Programming for Computational Control," *AIAA Journal*, Vol. 36, No. 2, 1998, pp. 219–226.

[8]Messac, A., "Control-Structure Integrated Design with Closed-Form Design Metrics Using Physical Programming," *AIAA Journal*, Vol. 36, No. 5, 1998, pp. 855–864.

[9]von Neumann, J., and Morgenstern, O., *Theory of Games and Economic Behavior*, Princeton Univ. Press, NJ, 1947.

[10]Hazelrigg, G. A., "On the Role and Use of Mathematical Models in Engineering Design," *Journal of Mechanical Design*, Vol. 121, No. 3, 1999, pp. 336–341.

[11]Cohon, J. L., "Multiobjective Programming and Planning," *Mathematics in Science and Engineering*, Vol. 140, Academic, San Diego, CA, 1978, p. 168.

[12]Stadler, S., "Multicriteria Optimization in Engineering and in the Sciences," *Mathematical Concepts and Methods in Science and Engineering*, edited by A. Miele, Plenum, New York, Vol. 37, 1988, pp. 120, 121.

[13]Thurston, D. L., Carnahan, J. V., and Liu, T., "Optimization of Design Utility," *Journal of Mechanical Design*, Vol. 116, No. 3, 1994, pp. 801–808.

[14]Osyczka, A., *Multicriterion Optimization in Engineering with Fortran Programs*, Ellis Horwood Series in Engineering, Halsted, New York, 1984, p. 48.

[15]Messac, A., and Turner, J. D., "Dual Structural Control Optimization of Large Space Structures," AIAA Paper 84-1042, May 1984.

[16]Messac, A., and Malek, K., "Control Structure Integrated Design," *AIAA Journal*, Vol. 30, No. 8, 1992, pp. 2124–2131.

[17]Sobieszczanski-Sobieski, J., James, B. B., and Dovi, A. R., "Structures Optimization by Multilevel Decomposition," *AIAA Journal*, Vol. 23, No. 11,

1985, pp. 1775–1782.
[18]Rao, S. S., Venkaya, V. B., and Khot, N. S., "Game Theory Approach for the Integrated Design of Structures and Controls," *AIAA Journal*, Vol. 26, No. 4, 1988, pp. 463–469.
[19]Chen, F. Y., and Li, D., "Multiobjective Optimization of Structures with and Without Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 2, 1996, pp. 392–397.
[20]Chen, W., Allen, J. K., Tsui, K. L., and Mistree, F., "A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors," *Journal of Mechanical Design*, Vol. 118, No. 4, 1996, pp. 478–485.
[21]Tappeta, R. V., and Renaud, J. E., "Multiobjective Collaborative Optimization," *Journal of Mechanical Design*, Vol. 119, No. 3, 1997, pp. 404–411.
[22]Messac, A., and Chen, W., "The Engineering Design Discipline: Is Its Confounding Lexicon Hindering Its Evolution?," *Proceedings of the Design Engineering Technical Conferences, Design Theory and Methodology*, American Society of Mechanical Engineers, Paper DETC98/DTM-5658, Atlanta, GA, Sept. 1998.
[23]MATLAB, The MathWorks, Inc., Natick, MA, 1984.

A. D. Belegundu
Associate Editor