Physical Programming: Effective Optimization for Computational Design

Achille Messac

Corresponding Author

Achille Messac, Ph.D.
Distinguished Professor and Department Chair
Mechanical and Aerospace Engineering
Syracuse University, 263 Link Hall
Syracuse, New York 13244, USA

Email: messac@syr.edu
Tel: (315) 443-2341
Fax: (315) 443-3099
https://messac.expressions.syr.edu/

Bibliographical Information

Messac, A., "Physical Programming: Effective Optimization for Computational Design," AIAA Journal, Vol. 34, No. 1, Jan. 1996, pp. 149-158. (Conference version was voted one of 12 best of 336 papers at McDonnell Douglas ASME best paper competition).

AIAA JOURNAL
Vol. 34, No. 1, January 1996

Physical Programming: Effective Optimization for Computational Design

Achille Messac*
Northeastern University, Boston, Massachusetts 02115

A new effective and computationally efficient approach for design optimization, hereby entitled physical programming, is developed. This new approach is intended to substantially reduce the computational intensity of large problems and to place the design process into a more flexible and natural framework. Knowledge of the desired attributes of the optimal design is judiciously exploited. For each attribute of interest to the designer (each criterion), regions are defined that delineate degrees of desirability: unacceptable, highly undesirable, undesirable, tolerable, desirable, and highly desirable. This approach completely eliminates the need for iterative weight setting, which is the object of the typical computational bottleneck in large design optimization problems. Two key advantages of physical programming are 1) once the designer's preferences are articulated, obtaining the corresponding optimal design is a noniterative process—in stark contrast to conventional weight-based methods and 2) it provides the means to reliably employ optimization with minimal prior knowledge thereof. The mathematical infrastructure that supports the physical programming design optimization framework is developed, and a numerical example provided. Physical programming is a new approach to realistic design optimization that may be appealing to the design engineer in an industrial setting.

I. Introduction

THE endeavor of computational design has witnessed significant evolution over the past decade. In the early eighties, the realization that the sequential treatment of structure and control failed to comprehensively recognize their interdependence prompted researchers to explore related integrated design approaches.[1–22] The area of multidisciplinary design optimization (MDO) has grown to the point of gaining near universal recognition in its ability to lead to better designs.[23] Unfortunately, the full power of MDO has only selectively been exploited. Government organizations and academia have developed and successfully employed MDO. At the same time, industry has generally failed to accept the quantum change in its product development infrastructure that would be required for the routine exploitation of MDO technology.

In addition to the disciplines of structures and control, MDO has been employed in conjunction with such diverse areas as trajectory determination, aeroelastic tailoring, and automobile active suspension design. Knowledge-based methods have also increasingly been used in the MDO context.[24] MDO-specific software is growing in numbers: (ASTROS,[25] ASCYNT,[26] and ENGINEOUS[27]). Other commercially available software have recently begun to incorporate MDO capabilities: PATRAN3, SDRC's IDEAS, ProENGINEER, and Intergraph are among such software. At the heart of the MDO problem is a multicriteria optimization problem. A number of books and other publications have been published on the subject. Ready applicability of such methods as utility theory to MDO is not within the current state of the art.[28,29] Reference 30 provides a broad perspective on the subject of design optimization.

It was mentioned earlier that industry has thus far underutilized MDO technology. The reasons for this situation are many.

1) The computational resources needed for the routine application of MDO are still beyond the reach of most organizations. For realistically complex designs, supercomputing, parallel computing, or distributed computing is needed if one is to keep computation time down to accustomed levels.

2) The use of full-featured analysis software tools within an optimization-based design environment remains an awkward process that often requires significant interface-coding efforts.

3) The application of optimization in the solution of large problems is too often an undertaking that frustrates all but the most experienced in the art. Without the benefit of significant experience on the part of the designer, the optimization problem formulation often fails to include important constraints. This omission usually leads the optimization process to converge to inappropriate design points.

4) The existence of several local minima, many of which might be highly undesirable, often poses severe difficulties.

5) The rigid framework for expressing design preferences can be considered singularly undesirable. That framework is typically limited to the use of such phrases as minimize C, need $A = B$, need $A < B$, and need $A > B$, where the quantities A, B, and C denote important attributes of the design. Moreover, the quantity C is usually an aggregate representation of the design preferences. The constituent components of this representation typically form a set of competing objectives. Methods for reconciling the competing nature of these objectives often require repeated optimization runs, thereby further exacerbating the computational nature of MDO. The latter point is discussed in detail later.

A. Problem Statement

To illustrate the conventional approach to optimization problem statements, consider the generic problem of designing a passenger car. The problem statement might read as follows: 1) Price must be less than $11,000. 2) Weight must be minimized as much as possible. 3) Mileage must be above 26 mpg and 18 mpg for highway and city driving, respectively. 4) Vibration response due to a given road disturbance must be below a given level. (5) ....

B. Shortcomings of Problem Statement

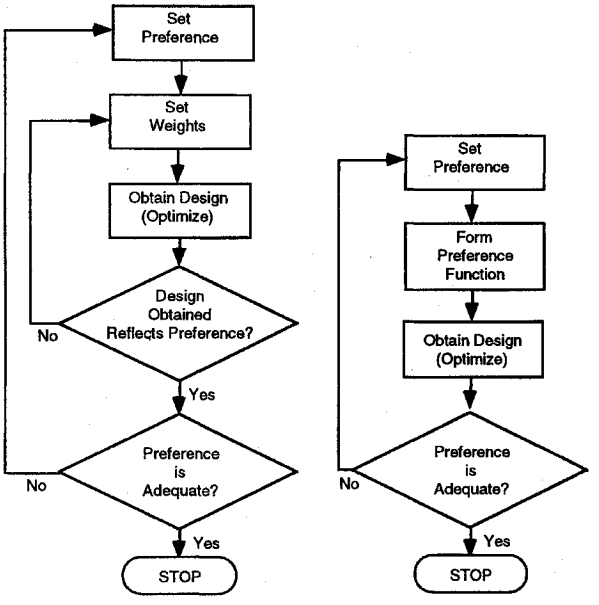
It is too inflexible, and it provides information within the context of a limited scenario. Because the solution is that of a narrowly posed problem, the designer must ask the critical question: How sensitive is the solution to both the design specifications and the design parameters? Insensitivity to a particular design specification suggests that the designer can relax that specification with little or no consequence. Strong sensitivity to a design parameter could alert the designer of possible fabrication problems regarding dimensional tolerance. The robustness properties of the design can become a concern.

C. Archimedian vs Preemptive Formulation Approaches

All real world design problems are implicitly or explicitly multiobjective, and approaches to finding the compromise (optimal)

Received Feb. 1, 1995; revision received July 28, 1995; accepted for publication July 28, 1995.

*Associate Professor, Department of Mechanical Engineering, Multidisciplinary Design Laboratory. Associate Fellow AIAA.



a) Weight-based approach b) Physical programming approach

Fig. 1 Design process: physical programming vs weight setting.

solutions are highly iterative. Archimedian and preemptive formulation approaches are briefly discussed in the following.

1) Weight-based (Archimedian) approaches are very difficult to implement in practice for realistic problems that comprise many weights. These methods typically require many iterations on the choice of weights and often provide no clear guidance as to how to converge to the right set of weights. The difficulties associated with weight-based methods are discussed in detail in Sec. II.A (Fig. 1). This process—which physical programming circumvents entirely—is particularly taxing to the designer's time and computational resources. The computational savings attributed to physical programming should be understood in that context.

Several methods are regarded as non-weight-based, in an explicit sense. A notable example is one where the Euclidean norm of the distance to the utopia point (the fictitious point that simultaneously optimizes all of the objectives) is minimized. However, even in the case of this seemingly sensible measure, weights must also be introduced to address both scaling and relative preference issues.

2) Preemptive approaches prioritize the various competing objectives. This prioritizing—ordering—of the objectives is an implicit statement that the priority-1 objective is infinitely more important than the priority-2 objective, that the priority-2 objective is infinitely more important than priority-3 objective, and so on. This prioritization allows the solution procedure to take a sequential approach: 1) optimize the priority-1 objective, 2) optimize the priority-2 objective subject to the priority-1 objective not getting worse, 3) and so on .... Some versions of multiobjective optimization combine the preemptive and Archimedian approaches.

The preemptive approach suffers from two major problems: 1) It is often very unrealistic to declare one objective infinitely more important than another. In the real world, preferences take the form of degrees. 2) There are $n!$ ways of ordering n objectives/criteria/constraints. Only a small number of possibilities can therefore be explored. Moreover, it may also happen that all of the solutions produced by a preemptive method (an ordering of the criteria) are unacceptable to the designer. It can easily be shown that a solution obtained by an Archimedian method is unobtainable by preemptive methods. For those reasons, weight-based methods have enjoyed significantly more popularity than the preemptive methods in the engineering design community.

In a general sense, if one is not using a preemptive method, then one is using a weight-based method that requires some choice of weight from the part of the designer, implicitly or explicitly. Even in the case of fuzzy optimization, the choice of membership functions can be interpreted as an implicit means of choosing weights. The choice of utility functions in utility theory, too, can be interpreted as a choice of weights. (Note that while utility theory clearly defines the properties of utility functions, it fails to provide practical means to develop such functions.) Physical programming is a method that requests physically motivated information from the part of the designer and produces a problem statement that reflects the realistic texture of the designer's preferences.

D. Objectives of This Paper

The following specific objectives are pursued in this paper: 1) develop a general framework for problem formulation that offers the flexibility required of real-world design problems (the framework allows for deliberate imprecision in the problem statement: a flexible infrastructure for expressing how good is good and how bad is bad), 2) develop a design process that aims to reduce the notoriously high MDO computational burden, and 3) ensure that the design process developed in objective 2 is readily compatible with existing optimization software.

This paper is organized as follows. The physical programming approach is presented in Sec. II. A numerical example is provided in Sec. III, and Sec. IV presents the conclusion.

Section II is divided into seven components. In Sec. II.A, the physical programming approach is compared with the conventional weight-based optimization approach. The superiority of the former is discussed from mathematical, algorithmic, and computational perspectives. Section II.B classifies the expression of preference under the physical programming methodology. In Sec. II.C, a method for the expression of intracriteria preference is developed, where an expansive design optimization lexicon is defined. This new lexicon brings both new flexibility and rigor to the design process by employing class functions. Section II.D develops the mathematical representation of the generic class functions. That development involves the creation of a new spline that possesses certain required properties. In Sec. II.E, the class function development algorithm is presented. Section II.F develops the aggregate preference function. The physical programming problem statement model is presented in Sec. II.G.

II. Physical Programming

A. Physical Programming vs Weight-Based Approach

1. Mathematical Relationship

This section develops the mathematical relationship that exists between the physical programming based preference function, J_{pf}, and the usual weighted-sum objective, J_w. This relationship will help establish the superiority of the former. The difficulty of expressing a general set of preferences using the weighted-sum approach is made evident.

The weight-based objective function is expressed as

$$J_w(\boldsymbol{g}) = \sum_i w_i[\boldsymbol{g}(\boldsymbol{x})]g_i(\boldsymbol{x}) \tag{1}$$

where $w_i[\boldsymbol{g}(\boldsymbol{x})]$ is the weight associated with the ith criterion g_i. (In many applications, the weight is chosen to be a constant.) In the preceding equation, the individual criteria are often squared before summation. That possibility does not materially impact the following discussion; the criterion is simply replaced by (or redefined as) its square. Note that g_i may represent objectives (criteria) that will be optimized or quantities that will be constrained.

The preference function based objective is written as

$$J_{pf}(\boldsymbol{g}) = J[\boldsymbol{g}(\boldsymbol{x})] \tag{2}$$

The preceding two expressions can be considered equivalent if they both have the same extrema. To establish that requirement, the first variation of each is evaluated:

$$\delta J_w(\boldsymbol{g}) = \sum_i p_i(\boldsymbol{g})\delta g_i \tag{3}$$

where

$$p_i(\boldsymbol{g}) = \sum_j \frac{\partial w_j}{\partial g_i} g_j + w_i \tag{4}$$

and

$$\delta J_{pf}(\boldsymbol{g}) = \sum_i q_i(\boldsymbol{g})\delta g_i \tag{5}$$

where

$$q_i(\boldsymbol{g}) = \frac{\partial J(\boldsymbol{g})}{\partial g_i} \tag{6}$$

Note that both J_w and J_{pf} reflect the same preference when the following differential equation is satisfied:

$$\sum_j \frac{\partial w_j}{\partial g_i} g_j + w_i = \frac{\partial J(\boldsymbol{g})}{\partial g_i} \tag{7}$$

or, in matrix form,

$$\left[\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{g}}\right]^T \boldsymbol{g} + \boldsymbol{w} = \frac{\partial J(\boldsymbol{g})}{\partial \boldsymbol{g}} \tag{8}$$

where $\boldsymbol{w}$ is a column vector whose ith entry is w_i. For the case where the minimum is at the boundary of a criterion domain—and the associated criterion variation does not vanish—the appropriate row in Eq. (8) is omitted.

Some useful observations can be made from the preceding development. Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be column vectors whose ith elements are denoted by p_i and q_i.

1) From Eqs. (3) and (5), note that (because the first variation of the preference function vanishes at the optimum) the vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ are parallel and are perpendicular to a hyperplane that is tangent to the hypersurface that borders the feasible hypervolume of the objective space (or criteria space): an intuitively satisfying geometrical interpretation of the preceding development.

2) In the case where each weight function depends only on the criterion it multiplies, and the preference function is a decoupled function of the criteria, Eq. (8) takes the form of m decoupled differential equations. Even in this simplified case, the correct weight is a complex function of the pertaining criterion.

3) In the common case where constant weight is assumed, Eq. (8) is further simplified, revealing the strict limitation associated with constant weights. The preference expressed can only be a linear function of the criteria [so that the right-hand-side of Eq. (7) can also be constant].

4) To express arbitrary preferences, the correct weight vector would have to be a complex function of the criterion vector.

5) The preference function, the criteria, and the weights are, in general, related by a set of coupled differential equations. The typical practice of assigning constant weights is therefore subject to gross inadequacies.

In light of the preceding observations, the author suggests that in the typical case where competing objectives are nonlinearly dependent on the design variables, iteratively guessing the appropriate values of the weights is—at best—a poor and inefficient way to engage in the design of complex systems.

Physical programming provides the means for direct expression of the preference, which fundamentally impacts the design process. Rather than expending substantial efforts tweaking weights and reoptimizing until a given set of preferences is achieved, the designer is allowed to concentrate more on the physical problem at hand and less on the art of converging to satisfactory weights. This issue is further discussed in the following. As is discussed in Ref. 28, several attempts at forming preference functions (utility functions) have been made. All have revealed the formidable difficulty of uncovering a general approach. This publication seeks to redress that situation.

2. *Generic Design Loops*

The following is a discussion of the design loops involved in the weight-based methods vs those of physical programming, where a general method for obtaining the preference function is available.

Figure 1a depicts the two generic loops involved in weight-based methods. First, the designer/decision maker expresses what he/she thinks the preferences should be. Based on those stated preferences (and possibly on initial analysis), the analyst chooses a set of weights and obtains the potentially optimal design. (This step is typically computationally intensive.) The next step is to assess whether the design obtained adequately reflects the stated preferences. Then, if the answer is no, choose new weights (inner loop); if yes, evaluate adequacy of preference statement (outer loop).

The basic deficiency of this approach is the existence of the inner-loop, the iterative weight setting, which constitutes a computational bottleneck for real (nonacademic) problems. Explicitly, the problem is twofold. First, obtaining a solution for each chosen set of weights may take hours to days of computer time for complex designs. Second, it may not be evident how to obtain an adequate convergence rate even for reasonably complex problems.

It is also important to observe that although the frequency of the inner loop is potentially high, that of the outer loop is typically not. The preference statement might be changed either because it was incorrect or because the designer wishes to explore other possibilities.

The preceding discussion suggests that the design optimization process would benefit greatly from two distinct developments: 1) the means for creating a mathematical function that directly expresses a priori preferences (removal of the inner loop described earlier) and 2) the ability to express preferences in fashions that are less than rigid and deliberately imprecise (as discussed in the sequel).

Physical programming is a new method that breaks new grounds on both counts.

B. Qualitative Classification of Preference

This section presents the physical programming method for expressing preference for each generic criterion. Preference generically falls under three classes, each comprising two cases.

As is depicted in Fig. 2, the preference classes are referred to as follows: 1) class-1: smaller-is-better (SIB), 2) class-2: larger-is-better (LIB), and 3) class-3: center-is-better (CIB).

For each class, the cases hard (H) and soft (S) are defined. Exactly what the shape of the preference function for the soft curves should be is the subject of later discussions.

These classes refer to the behavior of the preference function with respect to each generic criterion. For each criterion, a class function $\bar{g}_i$ is formed that constitutes a component of the aggregate preference function to be minimized. A lower value of the class function is considered better than a higher value. The utopian value of the class function is zero. For example: 1) If a criterion represents purchase price—to be minimized—class 1-S would apply. 2) If a criterion represents car mileage—to be maximized—class 2-S would apply. 3) In the cases where one only cares not to exceed some limits, the hard column would apply (Fig. 2).

It is important to observe that if the proper shape of the soft curves can be determined, the preceding classification offers significantly more flexibility than the typical weighted-criterion approach. Figure 3 depicts the limitations of the weighted-criterion approach. As can be seen, the weighted-criterion approach simply allows the designer to modify the slope of a straight line, whereas the class functions (Fig. 2) allow the preference function to be almost arbitrarily shaped. Section II.C develops the method for prescribing the appropriate shape of the class function.

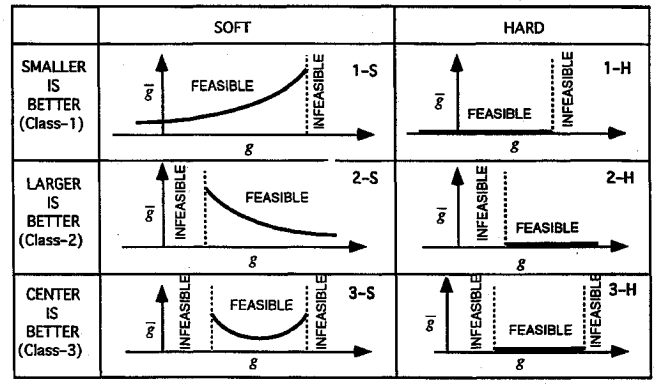


Fig. 2 Preference function classification in physical programming.

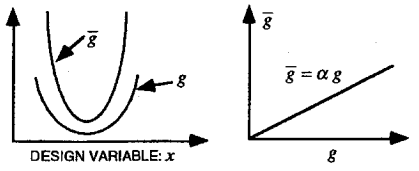


Fig. 3 Limitations of the weighted-criterion approach.

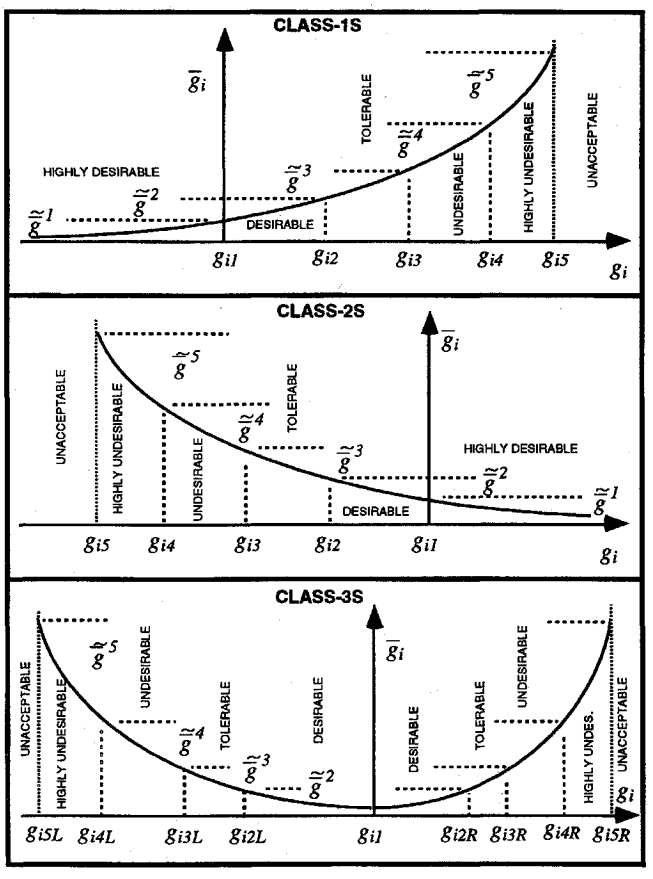


Fig. 4 Class function regions for *i*th generic criterion.

C. Intracriteria Preference

The effectiveness of physical programming rests substantially on its ability to create class functions that truly reflect the priorities of the designer. This section presents the methodical approach for quantitatively forming the generic class functions. This is accomplished by defining a physical programming lexicon that departs from conventional mathematical programming formalism. The development of mathematical expressions for the class functions is the subject of the next section.

1. *Physical Programming Lexicon*

Conventional mathematical formalism employs such terms as minimize A, maximize B, D less than E, F greater than G, or E equals F, where A–F denote quantities of interest in the design process. Physical programming explicitly recognizes the limitations of such a problem formulation framework by employing a new expansive and flexible lexicon. This lexicon comprises terms that characterize the degree of desirability of 6 ranges/domains/regions for each generic criterion for classes 1-S and 2-S, and 10 regions for class 3-S. To fix ideas, consider the case of class 1-S (Fig. 4); the situation for the other soft classes follows in a similar fashion.

The regions/ranges are defined as follows, in order of decreasing preference:

Unacceptable range (region 6) ($g_i \geq g_{i5}$): the range of values that the generic criterion may not take.

Highly Undesirable range (region 5) ($g_{i4} \leq g_i \leq g_{i5}$): an acceptable range that is highly undesirable.

Undesirable range (region 4) ($g_{i3} \leq g_i \leq g_{i4}$): an acceptable range that is undesirable.

Tolerable range (region 3) ($g_{i2} \leq g_i \leq g_{i3}$): an acceptable range that is tolerable.

Desirable range (region 2) ($g_{i1} \leq g_i \leq g_{i2}$): an acceptable range that is desirable.

Highly Desirable range (region 1) ($g_i \leq g_{i1}$): an acceptable range over which the improvement that results from further reduction of the criterion is desired but is of minimal additional value.

The preceding terms shall be understood in the following context.

2. *One vs Others Criteria Rule (OVO Rule)*

Consider the following two options concerning the class function values:

Option 1: Full reduction for one criterion across a given region (region k, $k = 3, 4, 5$).

Option 2: Full reduction for all of the other criteria across the next better region [region $(k-1)$].

Option 1 shall be preferred over option 2.

In other words, it is considered better for one criterion to travel across, say, the tolerable region than it is for all the other criteria to travel across the desirable region. [See inequality (11) later.] It is observed that, whereas the region limits define intracriterion preference, the OVO rule is a form of intercriteria preference.

3. *Qualitative Discussion*

First, consider the issues that are particular to a given criterion.

For each criterion, one must first define the class type. This is followed by the prescription of scalar values, for each criterion, that quantitatively define the criterion's preferred behavior via region limits (see Fig. 4). 1) For classes 1-H or 2-H, the designer must provide a single scalar value that defines the boundary of the feasible space. 2) For class 3-H, two values are needed. 3) For classes 1-S or 2-S, five values are required. 4) For classes 3-S, nine values are needed.

Second, the issues that are independent of both class type and criterion are discussed.

As is shown in Fig. 4, the value of the class function is the same at each of the region boundaries, regardless of class type or criterion (only the location of the boundary changes from criterion to criterion). As a consequence, as one travels across a given region type (say, tolerable), the change in the class function will always be of the same magnitude ($\tilde{\tilde{g}}^3$). This behavior of the class function values at the boundaries is the critical factor that makes each generic region type have the same numerical consequence for different criteria. This same behavior also has a normalizing effect, and results in favorable numerical conditioning properties. For example, consider the possible behavior of two distinct criteria over the tolerable region. The first criterion varies between 900 and 15,000, whereas the second varies between 1.5 and 3.8. Since both of those regions are tolerable, the respective class functions vertical excursion over that region is the same ($\tilde{\tilde{g}}^3$) regardless of the class type.

The property of the class function discussed earlier is expressed by the relation

$$\bar{g}_i(g_{ik}) \equiv \bar{\tilde{g}}^k \quad \forall i \qquad (1 \leq k \leq 5) \tag{9}$$

In light of the preceding discussion, it is noted that the change/improvement that takes place as one travels across the kth region reads $\tilde{\tilde{g}}^k$ (see Fig. 4).

The second feature of the class function that is independent of both the criterion under consideration and of the region type is as follows: To make 1) the highly undesirable region significantly worse than the undesirable region, 2) the undesirable region significantly worse than the tolerable region, 3) the tolerable region significantly worse than the desirable region, and 4) the desirable region somewhat worse than the highly desirable region, the following relations are enforced:

$$\tilde{\tilde{g}}^1 \text{ is a small positive number} \tag{10}$$

$$\tilde{\tilde{g}}^k > n_{sc}\tilde{\tilde{g}}^{k-1}, \qquad 2 \leq k \leq 5 \tag{11}$$

where

$$\tilde{\bar{g}}^k \equiv \bar{g}_i[g_{i(k)}] - \bar{g}_i[g_{i(k-1)}], \qquad 2 \le k \le 5 \tag{12}$$

$$\tilde{\bar{g}}^1 \equiv \bar{g}_i[g_{i(1)}] \tag{13}$$

and n_{sc} is the number of soft criteria. Note the absence of the index i in the left-hand side of Eqs. (12) and (13): The right-hand side is the same for all criteria. The inequality (11) reflects the OVO criteria rule.

The preceding discussion offers some similarities to the goal programming method, where a good value for each criterion is defined. That good value is closely associated with the g_{i1} value defined earlier. However, the similarities between goal programming and physical programming are largely limited to the preceding observation. The more comprehensive framework of physical programming could be of pivotal value to the practical designer. In a similar vein, many readers will observe a philosophical connection between physical programming and fuzzy optimization[32–33] where membership functions play a role similar to that of class functions in physical programming. A comprehensive discussion and analysis of the relative strengths of the two methods is beyond the scope of this publication.

D. Mathematical Representation of Generic Preference

This section develops an approach for developing the mathematical representation of the generic class functions. The class functions must have the following properties:

All soft classes
- (P1) strictly positive
- (P2) continuous first derivative
- (P3) strictly positive second derivative
- (P4) all properties must hold for arbitrary choices of region boundaries

Class 1-S
- (P5) strictly positive first derivative
- (P6) $\lim \bar{g}_i(g_i) = 0,\ g_i \to -\infty$

Class 2-S
- (P7) strictly negative first derivative
- (P8) $\lim \bar{g}_i(g_i) = 0,\ g_i \to \infty$

Class 3-S
- (P9) first derivative possesses one-and-only-one zero
- (P10) $(\partial \bar{g}_i/\partial g_i) = 0$ at $g_i = g_{i1}$

The justifications for the preceding properties are either obvious or provided in the next section. With these properties in mind, the objective is to develop a practical procedure for creating the generic class functions. Note that it is possible to remove the word strictly in the preceding properties in many instances by suffering only minor consequences.

In searching for a method to form the generic class function, the following observations are of interest. 1) Using a high-order polynomial with free coefficients (to be determined) is in general unacceptable, as this approach might easily violate property (P3), without extraordinary effort. 2) Conventional application of cubic spline methods is also unacceptable, as property (P3) might again be violated. The usual continuity of first and second derivative across spline segments does not necessarily satisfy property (P3).

To address the preceding difficulty, the necessary and sufficient conditions for the cubic spline method to lead to strictly positive curvature were developed [see Eqs. (34) and (35)]. Analysis of these conditions reveals that they are unduly restrictive. That is, the range of values of the criteria that lead to positive curvature for the class function is too limited. For that reason, a new spline is developed in the sequel, which offers significantly more flexibility. The successful development of such a spline is pivotal to the success of the class function development.

To fix ideas, the case of class 1-S is discussed first.

Observe that the class function regions can be represented by two generic functions (Fig. 4). The first can be represented by a decaying exponential, which is fully defined by its value and slope at a point (region 1). For regions 2, 3, 4, and 5 ($k = 2, 3, 4, 5$) the generic function takes the form of a spline segment that can be defined by its value and slope at its left and right boundaries.

The development of the spline segment that possesses a strictly positive second derivative begins by defining its second derivative in the form (for region k)

$$\frac{\partial^2 \bar{g}_i^k}{\partial g_i^{k2}} \equiv \left(\lambda_i^k\right)^2\left[a\left(\xi_i^k\right)^2 + b\left(\xi_i^k - 1\right)^2\right], \qquad 0 \le \xi_i^k \le 1 \tag{14}$$

where a and b are real, strictly positive constants, and where

$$\xi_i^k = \frac{g_i - g_{i(k-1)}}{g_{i(k)} - g_{i(k-1)}} \tag{15}$$

$$\lambda_i^k = g_{i(k)} - g_{i(k-1)} \tag{16}$$

It follows that the first derivative reads

$$\frac{\partial \bar{g}_i^k}{\partial g_i^k} = \left(\lambda_i^k\right)^3\left[\left(\frac{a}{3}\right)\left(\xi_i^k\right)^3 + \left(\frac{b}{3}\right)\left(\xi_i^k - 1\right)^3\right] + c \tag{17}$$

and the class function takes the form

$$\bar{g}_i^k = \left(\lambda_i^k\right)^4\left[(a/12)\left(\xi_i^k\right)^4 + (b/12)\left(\xi_i^k - 1\right)^4\right] + c\lambda_i^k\xi_i^k + d \tag{18}$$

Equation (18) possesses four constants a, b, c, and d that are determined by prescribing four quantities: the two displacements and slopes at the region boundaries (the algorithm for obtaining these boundary conditions is described in the following).

With the knowledge of the four constants a, b, c, and d, it is convenient to express the class function in region k as

$$\bar{g}_i^k = T_0\left(\xi_i^k\right)g_{i(k-1)} + T_1\left(\xi_i^k\right)g_{ik} + \tilde{T}_0\left(\xi_i^k; \lambda_i^k\right)s_{i(k-1)} + \tilde{T}_1\left(\xi_i^k; \lambda_i^k\right)s_{ik} \tag{19}$$

where $0 \le \xi_i^k \le 1, k = 2, \ldots, 5$, and

$$s_{ik} = \left.\frac{\partial \bar{g}_i}{\partial g_i}\right|_{g_i = g_{ik}} \tag{20}$$

$$T_0(\xi) \equiv \tfrac{1}{2}\xi^4 - \tfrac{1}{2}(\xi - 1)^4 - 2\xi + \tfrac{3}{2} \tag{21}$$

$$T_1(\xi) \equiv -\tfrac{1}{2}\xi^4 + \tfrac{1}{2}(\xi - 1)^4 + 2\xi - \tfrac{1}{2} \tag{22}$$

$$\tilde{T}_0(\xi, \lambda) \equiv \lambda\left[\tfrac{1}{8}\xi^4 - \tfrac{3}{8}(\xi - 1)^4 - \tfrac{1}{2}\xi + \tfrac{3}{8}\right] \tag{23}$$

$$\tilde{T}_1(\xi, \lambda) \equiv \lambda\left[\tfrac{3}{8}\xi^4 - \tfrac{1}{8}(\xi - 1)^4 - \tfrac{1}{2}\xi + \tfrac{1}{8}\right] \tag{24}$$

Equations (19–24) define the class function (class 1-S) for the ith criterion in the domain of the kth region ($k = 2, \ldots, 5$), if indeed the constants a and b are strictly positive. It is interesting to note the similarities between the preceding functions [Eqs. (21–24)] and the Hermite polynomials. The former more readily satisfy the present curvature requirements.

To establish the conditions under which this curvature positivity holds, observe from Eq. (14) that the constants a and b must be positive. To uncover the conditions under which those conditions are satisfied, a and b are evaluated as

$$a = \frac{3[3s_{ik} + s_{i(k-1)}] - 12\tilde{s}_i^k}{2\left(\lambda_i^k\right)^3} \tag{25}$$

and

$$b = \frac{12\tilde{s}_i^k - 3[s_{ik} + 3s_{i(k-1)}]}{2\left(\lambda_i^k\right)^3} \tag{26}$$

where

$$\tilde{s}_i^k = \tilde{\bar{g}}^k/\lambda_i^k \qquad (k = 2, 3, 4, 5) \tag{27}$$

Note that the variable $\tilde{s}_i^k$ is a characteristic slope for the kth region of the ith criterion. It comprises two components: the first ($\tilde{\bar{g}}^k$) represents the improvement that results from traveling across the kth region for any criterion; the second (λ_i^k) represents the length

of the same region for a given criterion. The first has a normalizing effect, whereas the second takes into account the preferred behavior of a given criterion.

Using Eqs. (25) and (26), it can be shown that the requirement that a and b be strictly positive is satisfied when the following two relations hold:

$$[s_{i(k-1)}]_{\min} \le s_{i(k-1)} \le [s_{i(k-1)}]_{\max}, \qquad k = 2, \ldots, 5 \tag{28}$$

$$(s_{ik})_{\min} \le s_{ik} \le (s_{ik})_{\max}, \qquad k = 2, \ldots, 5 \tag{29}$$

where

$$[s_{i(k-1)}]_{\min} = 4\tilde{s}_i^k - 3s_{ik}, \qquad [s_{i(k-1)}]_{\max} = \frac{4\tilde{s}_i^k - s_{ik}}{3} \tag{30}$$

$$(s_{ik})_{\min} = \frac{4\tilde{s}_i^k - s_{i(k-1)}}{3}, \qquad (s_{ik})_{\max} = 4\tilde{s}_i^k - 3s_{i(k-1)} \tag{31}$$

The preceding relations are used in the class function algorithm. They represent the constraints on the left boundary slope in terms of the right boundary slope [Eq. (28)] and vice versa [Eq. (29)]. To verify the effectiveness of this new spline, the magnitudes of the range of acceptability of the boundary slopes are evaluated and expressed as

$$\Delta s_{i(k-1)} = [s_{i(k-1)}]_{\max} - [s_{i(k-1)}]_{\min} = \tfrac{8}{3}\left(s_{ik} - \tilde{s}_i^k\right) \tag{32}$$

$$\Delta s_{ik} = (s_{ik})_{\min} - (s_{ik})_{\max} = \tfrac{8}{3}\left(\tilde{s}_i^k - s_{i(k-1)}\right) \tag{33}$$

The flexibility offered by this new spline is quantified by comparing these results with those obtained by the cubic spline method. By performing the same analytical development for the case of the cubic spline, one finds the magnitude of the range of acceptable slopes to be

$$\Delta s_{i(k-1)} = \tfrac{3}{2}\left(s_{ik} - \tilde{s}_i^k\right) \tag{34}$$

$$\Delta s_{ik} = \tfrac{3}{2}\left(\tilde{s}_i^k - s_{i(k-1)}\right) \tag{35}$$

Observe that the $\frac{8}{3}$ factor is replaced by $\frac{3}{2}$. Therefore, the new spline offers a 78% improvement in range of acceptable slope over the cubic spline. The author finds that this improvement is of great significance in the practical implementation of physical programming.

Finally, for region 1, the class function expression is represented by an exponential function. An acceptable mathematical expression reads

$$\bar{g}_i = g_{i1}\exp\left[\left(s_i^1/g_{i1}\right)(g_i - g_{i1})\right] \qquad \text{for} \qquad (g_i \le g_{i1}) \tag{36}$$

As stated earlier, the preceding discussion applies to class 1-S. The case of classes 2-S and 3-S follows readily from the previous discussion, with minor modifications. A few comments regarding those changes follow.

1) The case of class 2-S is the mirror-image of class 1-S.

2) The case of class 3-S requires the definition of nine values. These values need not possess any symmetry about the minimum of the class function. However, as discussed earlier, the class function vertical excursion across any given region is of the same magnitude for all classes (see Fig. 4).

E. Class Function Evaluation Algorithm

In Sec. II.D, the mathematical expressions for the generic class function are developed within the context of a single class function. For a given optimization/design problem, it is necessary to fully define all of the class functions before the aggregate preference function can be formed.

The following defines outlines of the algorithm for determining the class function value for a generic criterion value g_i:

Provide the input values:

i: criterion number, under consideration

g_i: value of ith criterion

g_{ij}: region boundaries (see Fig. 4)

n_{sc}: number of soft criteria

Evaluate

$$\bar{g}^1 = \tilde{\bar{g}}^1 = 0.1 \tag{37}$$

$$\tilde{\bar{g}}^k = \beta n_{sc}\tilde{\bar{g}}^{(k-1)} \qquad (2 \le k \le 5) \tag{38}$$

$$\bar{g}^k = \bar{g}^{(k-1)} + \tilde{\bar{g}}^k \qquad (2 \le k \le 5) \tag{39}$$

[See Eqs. (9), (12), and (13).]

$$\lambda_i^k = g_{i(k)} - g_{i(k-1)} \tag{40}$$

$$\tilde{s}_i^k = \tilde{\bar{g}}^k/\lambda_i^k \qquad (k = 2, 3, 4, 5) \tag{41}$$

(for class 1-S)

$$s_{i1} = \alpha\tilde{s}_i^2, \qquad 0 < \alpha < 1 \tag{42}$$

$$s_{ik} = (s_{ik})_{\min} + \alpha\Delta s_{ik} \qquad (k = 2, 3, 4, 5) \tag{43}$$

[See Eqs. (31) and (33).] Obtain $\bar{g}_i(g_i)$ using Eq. (19) or (36).

A few observations are in order: 1) Letting the constant α be small (<0.1) has the desired effect of maximizing the range of acceptable slopes at the region boundaries, that is, those that maintain a positive curvature. 2) The last two steps of the preceding algorithm apply to the class 1-S. A similar approach holds for the classes 2-S and 3-S. 3) The parameter β can be considered a convexity parameter. From the inequality (11), it is seen that β must be greater than 1. Furthermore, the value of β is used to enforce the convexity of all of the class functions as follows. After determining all of the class function values and slopes at all the region limits, it is verified that the constants a and b for each region are strictly positive using Eqs. (35) and (26). If this is not the case, β is increased, and new slopes and class function values are computed. It can be shown that this iterative process converges and does so quickly. In practice, starting with $\beta = 1.5$ and increasing by 0.5 works well.

F. Aggregate Preference Function Development

This section develops the mathematical representation of the aggregate preference function. To promote the robust implementation of physical programming, it is important to ensure that the form of the preference function is not responsible for the unintended introduction of new local minima into the problem. Gradient-based nonlinear programming codes would perform poorly in the solution phase. For the purpose of exploring areas of potential difficulties, the case of a single criterion with a single design parameter is discussed first. This is followed by the more general case.

1. One-Criterion with One-Parameter Case

In the case of a single criterion, with a single design variable, the preference function is assumed to simply involve the relevant class function. The quantities of interest are the design parameter x, the criterion expression that is governed by the physics of the problem g, and the class function that expresses preference $\bar{g}$.

Figure 5 depicts the various possibilities. In the first row (row 0), the behavior of the criterion function $g(x)$ is shown to comprise four possibilities: strictly increasing (si), strictly decreasing (sd), minimum (min), and maximum (max). The first column (column 0) depicts the three preference function possibilities. With the exclusion of the first row and column, the remaining entries describe the behavior of the function $\bar{g}(x)$, which is to be minimized. Figure 5 exposes the cases where $\bar{g}(x)$ possesses more than one minimum or no finite minimum.

The behavior of $\bar{g}(x)$ can be analyzed by evaluating the limits

$$\lim_{g \to g_0} \bar{g}(g), \qquad g_0 = \lim_{x \to \pm\infty} g(x) \tag{44}$$

and by examining the derivatives

$$\frac{\partial \bar{g}[g(x)]}{\partial x} = \frac{\partial \bar{g}}{\partial g}\frac{\partial g}{\partial x} \tag{45}$$

$$\frac{\partial^2 \bar{g}[g(x)]}{\partial x^2} = \frac{\partial^2 \bar{g}}{\partial g^2}\left(\frac{\partial g}{\partial x}\right)^2 + \frac{\partial \bar{g}}{\partial g}\frac{\partial^2 g}{\partial x^2} \tag{46}$$

Recall that $\partial^2\bar{g}/\partial x^2 > 0$.

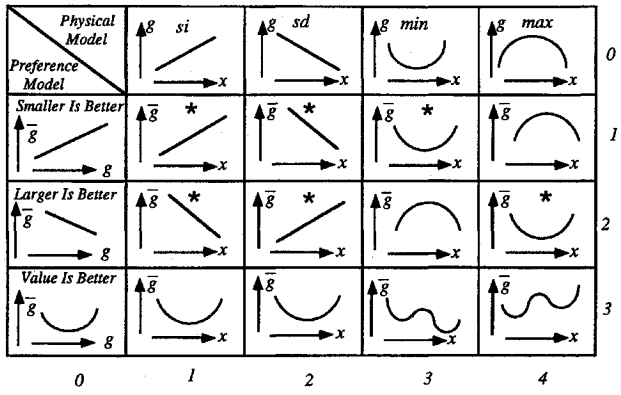


Fig. 5 Behavior of optimization model for the single-criterion single-parameter case.

The specific cases of Fig. 4 are discussed as follows, in light of Eqs. (44–46):

1) The entries (1, 1), (1, 2), (2, 1), and (2, 2) represent situations where constraints are necessary if the minimum is to be finite.

2) The entries (1, 3), (2, 4), (3, 1), and (3, 2) denote cases of a well-behaved function that possesses a single finite minimum.

3) The cases of entries (1, 4) and (2, 3) possess finite minima only when appropriate constraints are enforced. The potential for two minima, one at each endpoint, is unfortunately present.

4) The entries (3, 3) and (3, 4) represent troublesome situations, where a single-minimum or single-maximum criterion g is transformed into a two-minimum function $g(x)$ assuming no constraints.

It is important to note that the undesirable situations described in case 4 are not uncommon, as the weighted-sum-square method easily leads to those situations.

Further insight into the behavior of the preference function is obtained by analyzing the structure of the second derivative [Eq. (46)]. First, it is recalled that a function that has a strictly positive second derivative has at most one minimum.

Equation (46) shows that when $\partial^2 g/\partial x^2 > 0$, the quantity $\partial^2 \bar{g}/\partial x^2$ is strictly positive in the cases of entry (1, 1), (1, 2), and (1, 3). Equation (46) also shows that when $\partial^2 g/\partial x^2 < 0$, the quantity $\partial^2 \bar{g}/\partial x^2$ is strictly positive in the cases of entry (2, 1), (2, 2), and (2, 4). Moreover, from Eq. (46), it is seen that even in the case of entries (3, 1) and (3, 2), which possess single minima, there is no guarantee that $\partial^2 \bar{g}/\partial x^2 > 0$. By observing the first term of the right-hand side of Eq. (46), it is seen that the lack of guarantee for entries (3, 1) and (3, 2) can be alleviated by letting $\partial^2 \bar{g}/\partial g^2$ be strongly positive. (The algorithm for choosing the convexity parameter β exploits that fact.) The stars in Fig. 5 represent cases where the form of the preference function allows for the guarantee that $\partial^2 \bar{g}/\partial x^2 > 0$. As is shown in Sec. II.F.2, the said guarantee plays a key role in the formation of the aggregate preference function.

2. General Preference Function Formation

In this section, the method for forming the aggregate preference function is developed. The objective is to form a function that reflects the preferences expressed in each class function. The most direct approach is to simply add the individual class functions. But in doing so, it is important to ensure that the said addition does not lead to a multiple-minimum problem. Although this paper addresses the problem of multiparameter optimization, the following discussion follows the assumption of a single design parameter. The detailed analysis of the multiparameter case is the subject of ongoing studies. [Recall that the previous section discussed the number of minima of $\bar{g}(x)$, given those of $g(x)$.] This section discusses the number of minima of the sum $\boldsymbol{g} = \sum_i \bar{g}_i(x)$, given those of $\bar{g}(x)$.

Since the present interest is to determine when the sum $\boldsymbol{g}$ possesses a single minimum, it is important to examine the behavior of the second derivative of the said sum. (Again, it is recalled that a function that has strictly positive second derivative has at most one minimum.) In that light, it is observed that a sufficient condition for $\partial^2 \boldsymbol{g}/\partial x^2 > 0$ to hold is that $\partial^2 g_i/\partial x^2 > 0$ for at least one value of i, whereas $\partial^2 g_i/\partial x^2 \geq 0$ for the other values of i.

The cases where the condition $\partial^2 \bar{g}_i/\partial x^2 > 0$ holds have been discussed in the previous section. As was discussed in that section, the behavior of $\partial^2 \bar{g}_i/\partial x^2$ depends on that of $\partial^2 g_i/\partial x^2 > 0$, over which the designer has no control. Fortunately, the designer does have control over the quantity $\partial^2 \bar{g}_i/\partial g_i^2$. Equation (46) shows that the said flexibility can be exploited to make $\partial^2 \bar{g}_i/\partial x^2 > 0$, even when $\partial^2 \bar{g}_i/\partial x^2$ does not possess the desired property.

With the preceding comments in mind, the aggregate preference function is assumed to take the form

$$\hat{\boldsymbol{g}} = \frac{1}{n_{sc}} \sum_{i=1}^{n_{sc}} \bar{g}_i \tag{47}$$

For computationally motivated reasons, the actual function minimized is

$$\boldsymbol{g} = \log_{10} \hat{\boldsymbol{g}} \tag{48}$$

The quantities $\boldsymbol{g}$ and $\hat{\boldsymbol{g}}$ offer several computationally desirable features. 1) The scalar $\hat{\boldsymbol{g}}$ forces the optimization search path to be heavily governed by the criteria of the worst regions. This is largely due to enforcement of the relations Eqs. (11) and (38), which brings some mini-max features to the problem structure, where physically motivated regions govern the optimization path. 2) Because the quantity $\hat{\boldsymbol{g}}$ emphasizes the worst regions, it is possible to ignore the criteria that are in the much better regions and to periodically monitor the situation. As some of the ignored criteria become important, they are included. This exclusion/inclusion of selected criteria can be automated by periodically monitoring the values of the pertaining class functions. Since the computational intensity of the evaluation of the objective function is a critical factor in the optimization of large real-world problems, this economy of computation should not be ignored. 3) The taking of the logarithm in forming $\boldsymbol{g}$ has the desired result of mapping a domain that spans several orders of magnitude [largely because of Eqs. (11) and (38)] to one that typically involves one order of magnitude, $\hat{\boldsymbol{g}}$. This result is desired because most optimization codes work best when the values of the objective span less than two or three orders of magnitude. In all of the cases analyzed, the taking of the logarithm had the effect of reducing the number of iterations by at least 20% (while leading to the same optimum point).

The aggregate preference function is employed in the physical programming problem statement of the next section.

G. Physical Programming Problem Model

Building on the previous development, the physical programming problem statement takes the form

$$\min_{\boldsymbol{x}} g(\boldsymbol{x}) = \log_{10} \left\{ \frac{1}{n_{sc}} \sum_{i=1}^{n_{sc}} \bar{g}_i[g_i(\boldsymbol{x})] \right\} \qquad \text{(for soft classes)} \tag{49}$$

subject to

$$g_i(\boldsymbol{x}) \leq g_{i5} \qquad \text{(for class 1-S)} \tag{50}$$

$$g_i(\boldsymbol{x}) \geq g_{i5} \qquad \text{(for class 2-S)} \tag{51}$$

$$g_{i5L} \leq g_i(\boldsymbol{x}) \leq g_{i5R} \qquad \text{(for class 3-S)} \tag{52}$$

$$g_i(\boldsymbol{x}) \leq g_{iM} \qquad \text{(for class 1-H)} \tag{53}$$

$$g_i(\boldsymbol{x}) \geq g_{im} \qquad \text{(for class 2-H)} \tag{54}$$

$$g_{im} \leq g_i(\boldsymbol{x}) \leq g_{iM} \qquad \text{(for class 3-H)} \tag{55}$$

$$x_{jm} \leq x_j \leq x_{jM} \tag{56}$$

where g_{im}, g_{iM}, x_{jm}, and x_{jM} represent minimum and maximum values.

As can be seen, the preceding problem model conforms to the framework of most nonlinear programming codes, with possible minor rearrangements.

III. Example

A. Problem Definition

1. Plant Definition

The plant under consideration is idealized as a pinned-pinned sandwich beam that supports a motor (see Fig. 6). A vibratory disturbance (at v Hz) is imparted from the motor onto the beam, which is of length L, of width b, and symmetrical about its midplane. The variables d_1 and d_2, respectively, locate the contact of materials one and two, and two and three. The variable d_3 locates the top of the beam. The mass density, Young's modulus, and cost per unit volume for materials one, two, and three are, respectively, denoted by the triplets (ρ_1, E_1, c_1), (ρ_2, E_2, c_2), and (ρ_3, E_3, c_3). The mass of the motor is ignored in the following analysis.

In the design of the plant, the quantities of interest to the designer—or criteria—are as follows:

Fundamental frequency:

$$g_1 \equiv f_1 = (\pi/2L^2)(EI/\mu)^{\frac{1}{2}} \tag{57}$$

where

$$EI = (2b/3)\left[E_1 d_1^3 + E_2\left(d_2^3 - d_1^3\right) + E_3\left(d_3^3 - d_2^3\right)\right] \tag{58}$$

$$\mu = 2b[\rho_1 d_1 + \rho_2(d_2 - d_1) + \rho_3(d_3 - d_2)] \tag{59}$$

Cost:

$$g_2 \equiv c = 2b[c_1 d_1 + c_2(d_2 - d_1) + c_3(d_3 - d_2)] \tag{60}$$

Width:

$$g_3 \equiv b \tag{61}$$

Length:

$$g_4 \equiv L \tag{62}$$

Mass:

$$g_5 \equiv \mu L = m \tag{63}$$

Height (semiheight):

$$g_6 \equiv d_3 \tag{64}$$

Width of layer 1:

$$g_7 \equiv d_1 \tag{65}$$

Width of layer 2:

$$g_8 \equiv d_{21} = d_2 - d_1 \tag{66}$$

Width of layer 3:

$$g_9 \equiv d_{32} = d_3 - d_2 \tag{67}$$

The design parameters are

$$\boldsymbol{x} = \{d_1 \quad d_2 \quad d_3 \quad b \quad L\} \tag{68}$$

Note that design parameters can form a subset of the criteria if certain preferences concerning their values must be expressed.

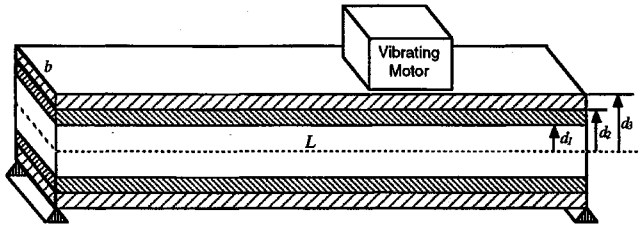


Fig. 6 Pinned-pinned sandwich beam design with vibrating motor.

2. Design Objectives and Problem Statement

The overall objective is to design the preceding sandwich beam in such a way as to passively minimize the vibration of the beam that results from the motor disturbance ($v = 10$ Hz). The various wishes and specifications are expressed by the designer/decision maker as follows. (It is not the intent of this paper to evaluate the wisdom of these preferences.)

Frequency: Since the motor disturbance is at 10 Hz, it is desired that the beam fundamental frequency be at least a decade higher, to obtain adequate passive attenuation. A fundamental frequency of between 150 and 200 Hz would be desirable. The smaller the better.

Cost: The smaller the better. A cost above \$2,000 is unacceptable. A cost of between \$1,000 and \$1,800 would be desirable. Up to \$1,900 would be tolerable, but above \$1,950 is highly undesirable.

Width: The beam must support some objects, so the wider the better. The beam must be at least 0.3 wide. A desirable value would range between 0.45 and 0.55 m. Below 0.35 m is highly undesirable. Above 0.40 would be tolerable. Above 0.55 m is minimally desired.

Length: The longer the better. The length must be at least 3 m long. The desirable length would range from 4 to 6 m. Between 3.8 and 4 would be tolerable, but below 3.3 would be highly undesirable. Above 6 m is minimally desired.

Mass: The lighter the better. The mass must be at most 2000 kg. Values ranging from 2700 to 2800 would be desirable. Below 2700 is minimally desired.

Semiheight: The smaller the better. A maximum value of 0.60 is required. The desirable range is between 0.3 and 0.4. Up to 0.5 is tolerable. Above 0.55 is highly undesirable. Below 0.3 is minimally desired.

Layer-1 thickness: Must be greater than 0.01 m.

Layer-2 thickness: Must be less than 0.01 m.

Layer-3 thickness: Must be less than 0.01 m.

The important issue here is to recognize that 1) the physical programming problem statement is ideally suited to address those real-world objectives in their true complexities, 2) the preceding flexibility in problem statement offered by physical programming is in stark contrast to the conventional approach to design problem statement, and 3) the new approach is clearly more reflective of real life. Solving the preceding relatively simple problem using the conventional weighted-sum approach would involve a highly iterative and uncertain weight tweaking endeavor. It is observed that the earlier problem statement involves a superset of the conventional problem formulation lexicon that is limited to the five terms: minimize, maximize, less than, greater than, and equal to. The author suggests that this rigid and limited framework is largely responsible for the unduly iterative nature of optimization (inner loop of Fig. 1a). In the case of large and complex problems, this highly iterative nature can be computationally prohibitive.

The preceding simple problem is chosen in an effort to introduce the new methodology without unduly obscuring its key features. In Ref. 33, Messac and Hattis address the more complex problem of the preliminary design of a simplified high-speed civil transport plane.

To perform the design optimization, the regions limits that delineate degrees of desirability are collected from the preceding objectives statements and reported in Table 1. The constants, design parameters, and design criteria values are shown in Table 2. The next section discusses the results.

The units that apply for Table 2 are as follows: E_i in Pascal, c_i in dollars per meter cubed, ρ_i in kilograms per meter cubed, frequency in hertz, mass in kilograms, and all distances in meters.

B. Results and Discussion

The optimal design results are presented in Tables 1–3. The first two present numerical results, whereas the third displays a qualitative visualization of the behavior of each criterion. In Table 3, a rightward arrow represents an improvement for the respective criterion. The frequency and the width experienced substantial improvements. The mass and the height showed moderate improvements. The cost remained in the desirable region without appreciable change. The length worsened but remained in the desirable region. The thicknesses of all of the layers remained in the acceptable region. However, the thickness of layer 2 reached its minimum, which points

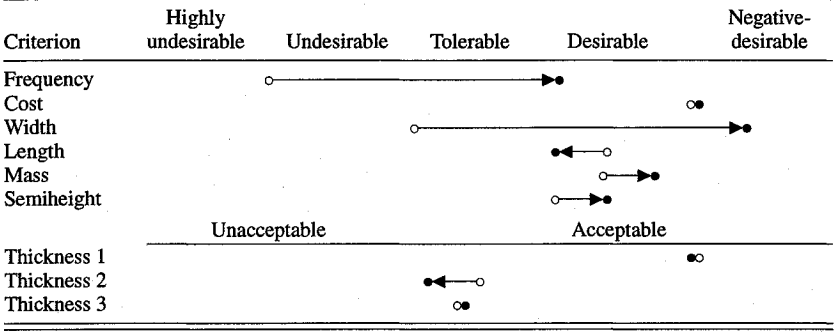
Table 1 Physical programming region limits table

		Highly undesirable	Undesirable	Tolerable	Desirable	Highly undesirable	
ith criter.	Class type	g_{i5}	g_{i4}	g_{i3}	g_{i2}	g_{i1}	
f_1	2-S	100	110	120	150	200	$+\infty$
c	1-S	2000	1950	1900	1800	1000	$-\infty$
b	2-S	0.30	0.35	0.40	0.45	0.55	$+\infty$
L	2-S	3.0	3.3	3.8	4.0	6.0	$+\infty$
m	1-S	2800	2700	2600	2500	2000	$-\infty$
d_3	1-S	0.60	0.55	0.5	0.4	0.3	$-\infty$
		Unacceptable				Acceptable	
d_1	2-H	—	—	—	—	0.01	$+\infty$
d_{21}	2-H	—	—	—	—	0.01	$+\infty$
d_{32}	2-H	—	—	—	—	0.01	$+\infty$

Table 2 Constants, design parameters, and criteria values

Constants	Value	Design parameters	Initial	Final	Criteria	Initial	Final
E_1	1.6e9	d_1	0.30	0.299	f_1	113	155.6
c_1	500	d_2	0.35	0.309	c	1060	1054
ρ_1	100	d_3	0.40	0.345	b	0.4	0.681
E_2	70e9	b	0.40	0.681	L	5	3.999
c_2	1500	L	5.00	3.999	m	2230	1845
ρ_2	2770	Preference function			d_3	0.4	0.345
E_3	200e9	g	3.541	1.074	d_1	0.3	0.299
c_3	800	$\hat{g}$	3473	11.86	d_{21}	0.05	0.010
ρ_3	7780	—	—	—	d_{32}	0.03	0.036

Table 3 Physical programming optimization results[a]



[a]○: initial point, ●: optimal point.

to the fact that it is of dubious utility in this particular design. Its material is relatively weak and expensive. When different input constants are used, the presence of the second layer can become important.

In a single optimization, the complex set of preferences are simultaneously satisfied. Of course, after examining the results, the designer may decide to explore other possibilities (outer loop of Fig. 1a). This deterministic iteration is entirely distinct from the typical highly iterative weight tweaking, which does not necessarily converge (inner loop of Fig. 1a).

It is noted that the computational and time savings attributed to physical programming cannot be quantified in the conventional manner; namely, 1) do an optimization run using physical programming, 2) do the same optimization run using a different method (method x), and 3) compare the CPU usage of the two. Under such a conventional comparison, the author would only expect physical programming to be highly competitive. But such a comparison would miss the point of this paper entirely. Such a comparison would implicitly assume that, given the designer's preference, method x provides the designer with the means to create the correct optimization problem statement—the solution of which would lead to the final optimal solution. Unfortunately, this is typically not the case. As previously discussed, one key distinctive feature of physical programming is its ability to formulate the problem statement that correctly reflects the designer's preference—the first time, and it does so in the background without the designer's intervention. This makes physical programming potentially appealing to industrial designers.

IV. Conclusion

This paper presented a novel approach (physical programming) to design optimization that offers two key advantages over conventional approaches. First, it offers a problem formulation and solution framework that conforms to real-life design, where an expansive new lexicon is employed. The terms desirable, tolerable, undesirable, and highly undesirable form the backbone of this new framework and bring a new flexibility, rigor, and deliberate imprecision to the design process. Reaching the optimal design becomes a less elusive and more deterministic process. Second, physical programming significantly impacts the computational burden by entirely eliminating the uncertain, time-consuming, and often frustrating process of weight tweaking. In the case of large and complex problems, this new approach may bring new possibilities. Physical programming is not a new approach to nonlinear programming; rather it is a new approach to using nonlinear programming for design optimization—a way that may be appealing to the design engineer in an industrial setting.

References

[1]Hale, A. L., Lisowski, R. J., and Dahl, W. E., "Optimizing Both the Structure and the Control of Maneuvering Flexible Spacecraft," *Proceedings of the AAS/AIAA Astrodynamics Conference* (Lake Placid, NY), 1983 (AAS Paper 83-377).

[2]Messac, A., and Turner, J. D., "Dual Structure-Control Optimization of Large Space Structures: Progress Report," NASA Symposium on Recent Experiences in Multidisciplinary Analysis and Optimization, Langley Research Center, NASA Conf. Publication 2327, Hampton, VA, April 1984, pp. 775–802.

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

[4]Messac, A., Turner, J. D., and Soosaar, K., "An Integrated Control and Minimum-Mass Structural Optimization Algorithm for Large Space Structures," *Proceedings of the Workshop on Identification and Control of Flexible Space Structures* (San Diego, CA), Vol. 2, JPL Publication 85-29, 1984, pp. 231–266.

[5]Gustafson, C. L., Aswani, M., Doran, A. L., and Tseng, G. T., "ISAAC (Integrated Structural Analysis and Control) via Continuum Modeling and Distributed Frequency Domain Design Technique," *Proceedings of the Workshop on Identification and Control of Flexible Space Structures*, Vol. 2, JPL Publication 85-29, 1985, pp. 28–310.

[6]Haftka, R. T., Martinovic, Z. N., Hallauer, W. L., and Schamel, G., "Sensitivity of Optimized Control Systems to Minor Structural Modifications," *Proceedings of the 26th Structures, Structural Dynamics, and Material Conference* (Orlando, FL), AIAA, New York, 1985, pp. 642–650 (AIAA Paper 85-0807).

[7]Messac, A., "Optimal Simultaneous Structural and Control Optimization of Large Space Structures," Doctoral Dissertation, Dept. of Aeronautics and Astronautics, Massachusetts Inst. of Technology, Cambridge, MA, Nov. 1985.

[8]Salama, M., Hamidi, M., and Demsetz, L., "Optimization of Controlled Structures," *Proceedings of the Workshop on Identification and Control of Flexible Structures*, Vol. 2, JPL Publication 85-29, 1985, pp. 311–328.

[9]Hale, A. L., "Integrated Structural/Control Synthesis via Set-Theoretic Methods," *Proceedings of the 26th Structures, Structural Dynamics, and Materials Conference* (Orlando, FL), AIAA, New York, 1985, pp. 636–641 (AIAA Paper 85-0806).

[10]Onoda, J., and Haftka, R. T., "An Approach to Structure/Control Simultaneous Optimization for Large Flexible Spacecraft," *AIAA Journal*, Vol. 25, No. 8, 1987, pp. 1133–1138.

[11]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.

[12]Belvin, W. K., and Park, K. C., "Structural Tailoring and Feedback Control Synthesis: An Interdisciplinary Approach," *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 3, 1990, pp. 424–429.

[13]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.

[14]Milman, M., Salama, M., Schad, R., Bruno, R., and Gibson, J. S., "Integrated Control-Structures Design: A Multiobjective Approach," Jet Propulsion Lab., JPL D6767, California Inst. of Technology, Pasadena, CA, Jan. 1990.

[15]Newsom, J. R., Layman, W. E., Waites, H. B., and Hayduk, R. J., "The NASA Controls-Structures Interaction Technology Program," 41st Congress of the International Astronautical Federation, Dresden, Germany, Oct. 1990.

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

[17]Padula, S. L., Sandwidge, C. A., Walsh, J. L., and Haftka, R. T., "Integrated Controls-Structures Optimization of a Large Space Structure," *Proceedings of the AIAA 31st Structures, Structural Dynamics, and Materials Conference* (Long Beach, CA), AIAA, Washington, DC, 1990, pp. 424–429 (AIAA Paper 90-1058).

[18]Maghami, P. G., Joshi, S. M., Walz, J. E., and Armstrong, E. S., "Integrated Controls-Structures Design Methodology Development for a Class of Flexible Spacecraft," 3rd Air Force/NASA Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, San Francisco, CA, Sept. 1990.

[19]Maghami, P. G., Joshi, S. M., and Lim, K. B., "Integrated Controls-Structures Design: A Practical Design Tool for Modern Spacecraft," *Proceedings of the 1991 American Control Conference* (Boston, MA), 1991.

[20]Padula, S. L., James, B. B., Graves, P. C., and Woodard, S. E., "Multidisciplinary Optimization of Controlled Space Structures with Global Sensitivity Equations," NASA TP-3130, Nov. 1991.

[21]Messac, A., and Turner, J. D., "Simultaneous Structure/Control Optimization of Complex Structures," AIAA/AAS Astrodynamics Conf., Williamsburg, VA, Aug. 1986.

[22]Messac, A., and Caswell, R., "CSID Control-Structure Integrated Design: Centralized vs Decentralized Control," AIAA Paper 92-1152, Feb. 1992.

[23]Anon., "Current State of the Art on Multidisciplinary Design Optimization," AIAA White Paper, AIAA, Washington, DC, Sept. 1991.

[24]Tong, S. S., and Powell, D., "Integration of Artificial Intelligence and Numerical Optimization Techniques for the Design of Complex Aerospace Systems," AIAA Paper 92-1189, Feb. 1992.

[25]Anon., ASTROS, Wright Research and Development Center, Flight Dynamics Lab., Wright–Patterson AFB, OH, Dec. 1988.

[26]Anon., ACSYNT, ACSYINT Inst., Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, 1994.

[27]Anon., ENGINEOUS, GE Corporate Research and Development, 1994 (not commercial).

[28]Cohon, J. L., "Multiobjective Programming and Planning," *Mathematics in Science and Engineering*, Vol. 140, Academic, San Diego, CA, 1978, pp. 163–179.

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

[30]Haug, E. J., and Arora, J. S., *Applied Optimal Design*, Wiley-Interscience, New York, 1979.

[31]Fedrizzi, M., Kacprzyk, K., and Roubens, M., "Interactive Fuzzy Optimization," *Lecture Notes in Economics and Mathematical Systems*, Vol. 368, Springer–Verlag, New York, 1991.

[32]Lai, Y.-J., and Hwang, C.-L. (eds.), "Fuzzy Mathematical Programming, Methods and Applications," *Lecture Notes in Economics and Mathematical Systems*, Vol. 394, Springer–Verlag, 1992.

[33]Messac, A., and Hattis, P., "High Speed Civil Transport (HSCT) Plane Design Using Physical Programming," *Proceedings of the AIAA 36th SDM Conference* (New Orleans, LA), AIAA, Washington, DC, 1995 (AIAA Paper 95-1401).