Physical Programming Design Optimization for High Speed Civil Transport

Achille Messac*
Northeastern University, Boston, Massachusetts 02115
and
Philip D. Hattis†
Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts 02139

Introduction

THIS Note applies the new design optimization methodology entitled physical programming (Messac[1]) to the preliminary design of the High Speed Civil Transport (HSCT) plane. The HSCT plane design offers a unique combination of technical challenges. The disciplines of structural dynamics, propulsion, thermodynamics, aerodynamics, and fluid dynamics adversely interact with such equally important issues as environmental pollution, noise, and economic competitiveness. It is shown that this multidisciplinary design effort could benefit from a comprehensive application of physical programming. The detailed analytical development that forms the basis of this Note is presented by Messac and Hattis.[2] The next section presents an introduction to the physical programming method. This is followed by a discussion of the HSCT model and its use in an HSCT preliminary design using physical programming. The last section provides reflective remarks on the applicability of physical programming to a large class of system design problems.

Physical Programming

Motivation for the Development of Physical Programming

The application of physical programming in design optimization offers distinct advantages: reduced design time, reduced computational effort, and enhanced ability to obtain an optimal design. Physical programming employs a flexible and natural framework for decision making and problem formulation. It possesses the unique and desirable feature that the designer does not need to specify optimization weights in the problem-formulation phase. Rather, the designer specifies ranges of differing degrees of desirability for each design metric. Under the physical programming methodology, the designer expresses the objectives regarding each design metric in a deliberately imprecise manner.

Physical programming departs from conventional design problem formulation paradigms. Rather than making the real-world problem statement conform to the limitations of conventional mathematical formalisms, the designer is free to express the design objectives without regard to the intricacies of the supporting mathematical infrastructure. Specifically, the designer is not forced to cast the problem into the conventional lexicographic (pre-emptive) or Archimedean formalisms. The former forces the designer to declare some issues as being infinitely more important than others, whereas the latter requires the designer to form an aggregate objective function, the morphology of which usually involves weights, which are singularly difficult to determine correctly. Physical programming[1] is shown to retain the desirable features of both formalisms, while avoiding their respective limitations. It does not declare any one issue infinitely more important than another (except for constraints). Nor does it require the designer to provide weights in the formation of the aggregate objective. The formation of the aggregate objective is instead guided by the explicitly stated wishes of the designer.

To see the flexibility of the physical programming problem formulation, consider one of the design metrics of the HSCT problem, recurring-cost-per-passengers-seat c_r. If a multiobjective HSCT mathematical model is available, and we wish to minimize c_r (among other objectives), two options might come to mind. In the first, c_r would become part of an aggregate objective function with an associated multiplicative weight. In the second option, an inequality constraint would be invoked, viz., $c_r < 20$ dollars. Both options fail to adequately exploit the designer's true wishes and his/her expert knowledge of the problem. If c_r is above 25, it is very important that it be minimized. If on the other hand, c_r is less than 15, it is less important to further minimize it at the expense of other objectives. A constant weight cannot model that situation. The inequality constraint approach, too, is ineffective since in all likelihood the manager would not find 20.01 dollars unacceptable (which the inequality constraint rules out as a potential solution).

The realistic statement of preference for c_r might take the form:

Highly desirable:	<17 (dollars)
Desirable:	17 to 21
Tolerable:	21 to 22
Undesirable:	22 to 23
Highly undesirable:	23 to 24
Unacceptable:	>24

Physical programming allows for that realistic description of the problem. We note that this description encompasses both hard objectives (unacceptable), and soft objectives (highly desirable to highly undesirable), which are expressed in terms of ranges of differing degrees of desirability.

The physical programming method is philosophically connected to several other optimization methods that explicitly exploit expert knowledge of the physical problem, or allow for the use of imprecise information. The more notable ones are as follows. The Taguchi method[3] judiciously selects and performs experimental trials to determine nominal values for the design parameters in the face of uncertainties (noise factors). In a different vein, the goal programming method[4] is based on mathematical programming. Its key feature is its ability to seek a physically meaningful value (goal) for each design metric. The method of inequalities[5] uses inequalities for each design metric. The basic procedure is to start with a large feasible objective space, and to progressively decrease the feasible space by moving the inequality boundaries until a satisfactory solution is obtained. Utility theory[6] 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). In recent years, the fuzzy set theory (FST) (Ref. 7) has been used with celebrated success. FST correctly recognizes that deliberate imprecision in decision making is desirable. It allows for the linguistic manipulation of conditions (e.g., if velocity is low, then force is high). FST provides the means to manipulate imprecision within a rigorous framework.

Since the early 1980s, interest in the area of engineering design in a multidisciplinary setting has grown.[8-11] The publications just cited focus on the area of control-structure-integrated-design (CSID), which is computationally intensive. Be-

Received March 15, 1995; revision received Nov. 10, 1995; accepted for publication Nov. 18, 1995. Copyright © 1996 by A. Messac and P. D. Hattis. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

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

†Principal Member of the Technical Staff. Associate Fellow AIAA.

cause of the computational intensity of the CSID problem, it is very costly to optimize (and reoptimize) using an incorrect objective function, which 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 aggregate objective function that correctly reflects the full texture of the decision-maker's wishes.

Physical Programming Application Synopsis

This section describes the procedure for applying physical programming. The current physical programming implementation is embodied in the software package entitled PhysPro (Ref. 12), which is Matlab-based and at a more advanced stage than the version described in Ref. 1. To apply physical programming to any problem, several well-defined steps are performed.

We begin with some requisite preliminary discussions and definitions. To perform the design, it is assumed that the designer has the ability to alter a set of system features, design variables, represented by the vector $\boldsymbol{x} = (x_1, x_2, \ldots)$. The problem formulation also involves identifying those characteristics of the system, design metrics, that allow the designer to judge the effectiveness of the outcome. Those characteristics are denoted by the vector $\boldsymbol{g} = (g_1, g_2, \ldots)$. Design metrics may be quantities that the designer wishes to minimize, maximize, take on a certain value (goal), fall in a particular range, or be less than, greater than, or equal to particular values.

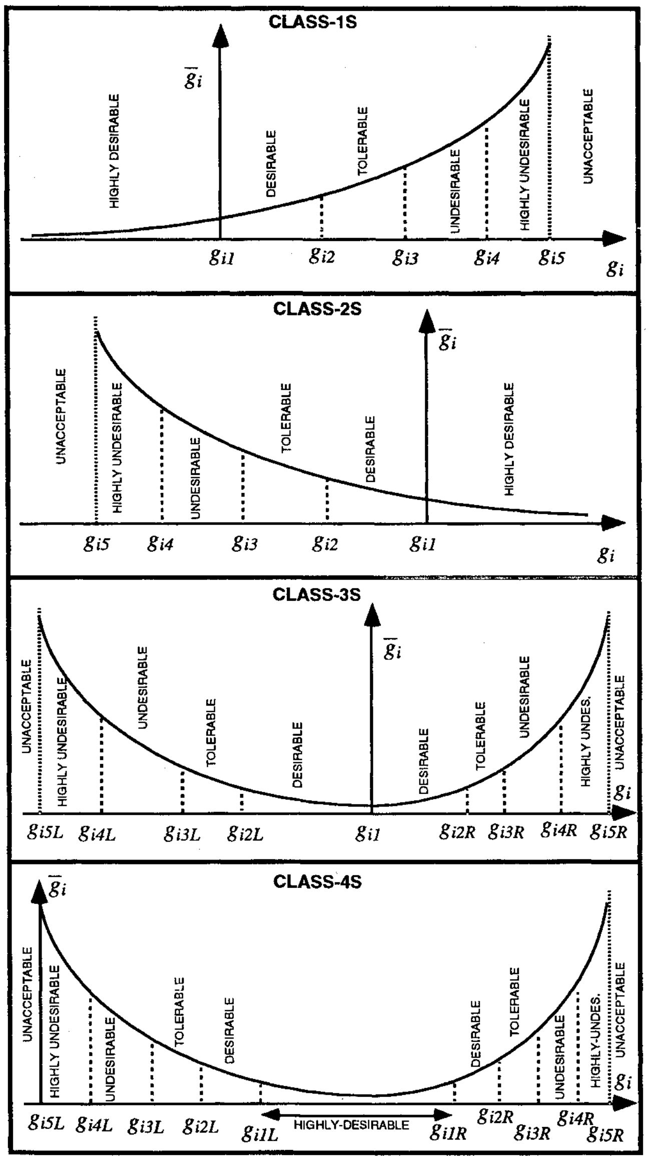

Fig. 1 Class functions regions for *i*th generic design metric.

Classification of Objectives and Class Functions

Within the physical programming procedure, the designer expresses objectives with respect to each metric using four different classes. Figure 1 depicts the qualitative meaning of each class. The value of the metric under consideration g_i is on the horizontal axis, and the function that will be minimized for that metric $\bar{g}_i$, hereby called the class function, is on the vertical axis. Each class comprises two cases, hard and soft, referring to the sharpness of the preference. All soft class functions will become constituent components of the aggregate objective function (that will be minimized). The desired behavior of a generic metric is described by one of eight subclasses, four soft and four hard. These classes, illustrated in Fig. 1, are defined as follows:

Soft:

Class-1S	smaller-is-better, i.e., minimization
Class-2S	larger-is-better, i.e., maximization
Class-3S	value-is-better
Class-4S	range-is-better

Hard:

Class-1H	must be smaller, i.e., $g_i \le g_{i_{\max}}$
Class-2H	must be larger, i.e., $g_i \ge g_{i_{\min}}$
Class-3H	must be equal, i.e., $g_i = g_{i_{\mathrm{val}}}$
Class-4H	must be in range, i.e., $g_{i_{\min}} \le g_i \le g_{i_{\max}}$

The class functions, shown in Fig. 1, provide the means for a designer to express ranges of differing levels of preferences for each design metric, resulting in a deliberately imprecise framework. Next, we explain how quantitative specifications are associated with each performance metric.

Physical Programming Lexicon

From the previous discussion, we note that physical programming allows the designer to express preferences with regard to each design metric with more specificity than by simply using the terms minimize, maximize, greater than, less than, or equal to. The physical programming lexicon comprises terms that characterize the degree of desirability of six ranges for each generic design metric for classes 1S and 2S, 10 for class 3S, and 11 for class 4S. To illustrate, consider the case of class 1S, shown in Fig. 1. The ranges are defined as follows, in order of decreasing preference:

1) Highly desirable range ($g_i \le g_{i1}$): an acceptable range over which the improvement that results from further reduction of the performance metric is desired, but is of minimal additional value.

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

3) Tolerable range ($g_{i2} \le g_i \le g_{i3}$): an acceptable, tolerable range.

4) Undesirable range ($g_{i3} \le g_i \le g_{i4}$): a range that, while acceptable, is undesirable.

5) Highly undesirable range ($g_{i4} \le g_i \le g_{i5}$): a range that, while still acceptable, is highly undesirable.

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

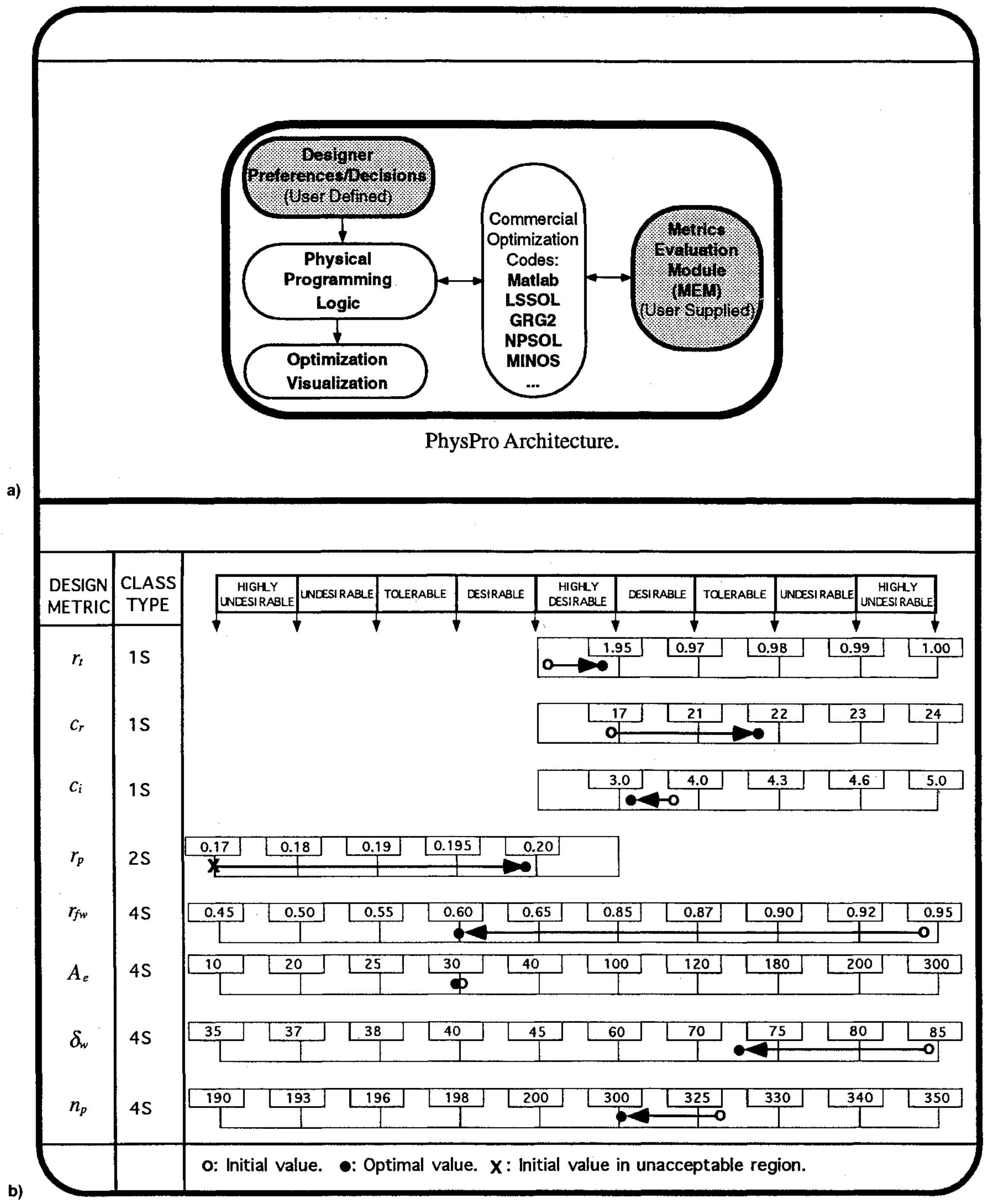
Designer Preferences/Decisions (User Defined)
Physical Programming Logic
Optimization Visualization
Commercial Optimization Codes: Matlab LSSOL GRG2 NPSOL MINOS ...
Metrics Evaluation Module (MEM) (User Supplied)
PhysPro Architecture.
a)
DESIGN METRIC
CLASS TYPE
HIGHLY UNDESIRABLE
UNDESIRABLE
TOLERABLE
DESIRABLE
HIGHLY DESIRABLE
DESIRABLE
TOLERABLE
UNDESIRABLE
HIGHLY UNDESIRABLE
r_t 1S 1.95 0.97 0.98 0.99 1.00
c_r 1S 17 21 22 23 24
c_i 1S 3.0 4.0 4.3 4.6 5.0
r_p 2S 0.17 0.18 0.19 0.195 0.20
r_{fw} 4S 0.45 0.50 0.55 0.60 0.65 0.85 0.87 0.90 0.92 0.95
A_e 4S 10 20 25 30 40 100 120 180 200 300
δ_w 4S 35 37 38 40 45 60 70 75 80 85
n_p 4S 190 193 196 198 200 300 325 330 340 350
o: Initial value. ●: Optimal value. X: Initial value in unacceptable region.
b)

Fig. 2 Physical programming software and optimization results.

The parameters $g_{i1}-g_{i5}$ are physically meaningful values that are specified by the designer to quantify performance objectives associated with the ith design metric. These parameters delineate the desirability ranges within each metric. Reference 1 discusses the quantitative mathematical implications of the previous definitions and provides the mathematical means for the development of the class function for each generic metric, in a multiobjective setting. We will simply note here that these definitions entail an implied intracriteria and intercriteria preference.

Physical Programming Application Steps

We now discuss the procedure for applying physical programming, which involves four distinct steps.

Step 1: create a software module [metrics-evaluation-module (MEM)] that uses the current numerical values of the design parameters x as input, and that provides the corresponding numerical values of the design metric vector g, as output (i.e., a software module that describes the behavior of the physical system). This module is shown in the PhysPro Architecture (Fig. 2a). In the present case, since PhysPro is implemented in the Matlab environment, the MEM development can exploit the full power and versatility of Matlab, or can also be a C or Fortran module that interfaces with Matlab, using the Matlab–MEX facilities. It can also incorporate structural analysis codes as was done in Ref. 11, or other intradisciplinary software.

Step 2: prescribe the class-type for each metric (1S–4H) (see Fig. 1). (This is done within one of the PhysPro Windows.)

Step 3: define the range–limits for each design metric (Figs. 1 and 2b). The designer prescribes 5 limits for class 1S or 2S, 9 for 3S, and 10 for 4S. For the hard classes, the designer prescribes one limit for class 1H, 2H, or 3H, and two limits for 4H.

Step 4: Solve the constrained minimization problem that is defined by the physical programming model.[1] 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 the uncertain art of tweaking weights, the designer's time is constructively spent exploring the implications of the various physically meaningful preference choices.

HSCT Model

Background

The economics of an operational HSCT must be sufficiently favorable for airlines to be interested in the operation of a large fleet. The major economic factors include per-vehicle production costs and incremental operational expenses. Generally, the production costs of airplanes have a direct correlation with the dry vehicle mass. Operational expenses for long-haul aircraft are dominated by fuel price, which is proportional to fuel consumption. However, operational expense also includes other factors such as staffing the vehicle with a flight crew. Performing an HSCT analysis with respect to the dominant economic factors for a maximum range mission will assure a design solution with the most reasonable economics relative to other intercontinental flight vehicles (such as existing subsonic jumbo jets). Consideration of HSCT-required wet–mass (a combination of the dry mass and the mission fuel requirement) is also a suitable design metric.

The HSCT cruise phase assumes that the cruise will be at Mach 2.5, at an altitude of 55,000 ft, with a total cruise range of 6200 mile in support of a 7000-mile flight from airport to airport. To support the demonstration of the HSCT design, an HSCT parametric formulation is constructed, then optimized. The analytical development of the HSCT governing equations is presented in Ref. 2. The remainder of this section discusses 1) the design parameters, 2) the design metrics, 3) the designer preferences, and 4) the optimal design obtained using physical programming.

HSCT Design Parameters

The design parameters of the simplified HSCT model have been selected to allow considerable vehicle design flexibility in the primary configurational and performance characteristics. The design parameters are x_1, engine inlet area A_e; x_2, wingspan s_w; x_3, wing sweep-back angle δ_w; x_4, number of passengers (capacity) of the vehicle n_p; and x_5, propellant-tank volume V_t.

HSCT Design Metrics

The goodness of a particular design is assessed in terms of a set of design metrics. These design metrics are g_1, tank volume-ratio r_t (actual to maximum allowed); g_2, recurring cost per passenger seat c_r; g_3, initial cost per passenger seat c_i; g_4, propellant mass ratio (final to initial) r_p; g_5, fuselage-length/wing-root-length ratio r_{fw}; g_6, engine inlet area A_e; g_7, wing sweep-back angle δ_w; and g_8, number of passengers n_p.

Note that three quantities, A_e, δ_w, and n_p, appear both as design parameters and design metrics. This situation is reflective of the fact that, while these quantities are allowed to vary in the search for the optimal design, the designer also has the ability to express explicit preferences regarding their values. This realization is easily exploited in physical programming.

HSCT Design Problem: Expression of Preferences

The physical programming HSCT design problem is articulated by simply expressing the designer's preference in terms of the physical programming lexicon as described previously. The problem is completely defined once the designer chooses the class type for each design metric (step 1), and specifies the region limits for each design metric (step 2). The chosen classes are as follows: r_t, class 1S; c_r, class 1S; c_i, class 1S; r_p, class 2S; r_{fw}, class 4S; A_e, class 4S; δ_w, class 4S; and n_p, class 4S. The specification of the associated range limits is given in Fig. 2b (step 2).

Optimal Design Obtained Using Physical Programming

Using the information from steps 1 and 2 as inputs to the physical programming software, PhysPro, the optimal design is obtained and depicted in Fig. 2b, where the arrows describe the optimization process. As can be seen, for the initial design, one design metric is unacceptable, two are highly undesirable, and the others are tolerable or better. For the final design, all of the metrics are tolerable or better. Examination of Fig. 2b offers some understanding of the trades that took place. The details of the HSCT design model and optimization are provided in Ref. 2.

Conclusions

This Note presents the application of physical programming to the HSCT plane design problem. It shows that physical programming is a potentially powerful tool in the design of complex systems, making it possible 1) to express a complex set of preferences; 2) to obtain a design that truly reflects the designer's preferences; 3) to complete the design process with a significantly reduced computational burden; and 4) to avoid the necessity of choosing the right weights, a process that can be extremely difficult. This publication demonstrates the flexibility of physical programming in cases involving highly nonlinear governing dynamics and highly competing objectives of disparate nature.

References

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

[2]Messac, A., and Hattis, P. D., "High Speed Civil Transport (HSCT) Plane Design Using Physical Programming," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 36th Structures, Structural Dynamics, and Materials Conference* (New Orleans, LA), AIAA, Washington, DC, 1995, pp. 2018–2031 (AIAA Paper 95-1401).

[3]Byrne, D. M., and Taguchi, S., "The Taguchi Approach to Parameter Design," *Quality Congress Transaction—Anaheim*, American Society for Quality Control, May 1986, pp. 168–177.

[4]Ignizio, J. P., *Goal Programming and Extensions*, Lexington Books, D. C. Heath and Co., MA, 1976.

[5]Zakian, V., "New Formulation for the Method of Inequalities," *Proceedings of the IEEE*, Vol. 126, 1979, pp. 579–584.

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

[7]Zadeh, L. A., "Fuzzy Sets as a Basis for the Theory of Possibility," *Fuzzy Sets and Systems*, Vol. 1, 1978, pp. 3–28.

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

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

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

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

[12]Messac, A., "PhysPro: Software Package for Optimal Design," Multidisciplinary Design Lab., Northeastern Univ., Boston, MA, 1994.