Development of a multidisciplinary design optimization test simulator

K.F. Hulme and C.L. Bloebaum
Multidisciplinary Optimization and Design Engineering Laboratory (MODEL), Department of Mechanical and Aerospace Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA

Abstract Many of the method development efforts in the field of multidisciplinary design optimization (MDO) attempt to simplify the design of a large, complex system by dividing the system into a series of smaller, simpler, and coupled subsystems. A representative and efficient means of determining the feasibility and robustness of MDO methods is crucial. This paper describes the construct and applications of a test simulator, CASCADE (Complex Application Simulator for the Creation of Analytical Design Equations), that is capable of randomly generating and then converging a system of coupled analytical equations, of user-specified size (Hulme and Bloebaum 1996). CASCADE-generated systems can be used for test sequencing and system reduction strategies, convergence strategies, optimization techniques, MDO methods, and distributed computing techniques (via Parallel Virtual Machine), among others.

1 Background

It is desired to achieve a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. The interaction of all participating engineering groups throughout the design cycle is a truly multidisciplinary effort. Many of the recently developed capabilities to address concurrent design have stemmed from the field of multidisciplinary design optimization (MDO). The MDO approach is intuitive in that one often attempts to decompose one large task into a group of smaller, interrelated (coupled), and more manageable tasks (Sobieski 1982). The large task is often referred to as a system, and the smaller, interrelated tasks are called subsystems. Each subsystem contains design variables, as well as additional unknown outputs, often referred to as behaviour variables. These subsystem variables (either design or behaviour) are collectively referred to as modules of the system. Each subsystem can be thought of as participating design group of a large scale design. An example of this would be the aerodynamics division of the design of an aircraft. One goal of MDO is to analyse these subsystems concurrently, thus speeding up the design time of the overall product. This method was first established by applying a linear decomposition method to a hierarchical (top-down) system (Sobieski 1982).

Most design cycles contain participating groups that interact laterally. Such design cycles are thus nonhierarchical in nature. The global sensitivity equation (GSE) method was the first approach to extend the concepts of the linear decomposition method to nonhierarchical systems (Sobieski 1982; Bloebaum 1989). This method uses local sensitivities (derivatives that are computed within each subsystem) to compute total system sensitivities (Hajela et al. 1990). The concurrent subspace optimization (CSSO) procedure (Sobieski 1990) allocates the system design variables to subspaces, which correspond to separate engineering disciplines. Each subspace performs a separate optimization by operating on its own unique set of design variables. The coordination problem is solved by using the GSE. To aid in convergence, heuristic approaches have been developed to adjust move limits and related parameters (Bloebaum 1991). A related approach, called collaborative optimization (CO) (Koo 1995), decomposes the problem further and eliminates the need for separate system and sensitivity analyses. This is accomplished by combining the design variables and those state variables that couple the subspaces into one vector. These system-level variables may be shared between subspaces, and hence may assume nonunique values.

An MDO issue that is the focus of much research today is the concept of sequencing the coupled subsystems. Sequencing is a methodology that reorders the modules in a given system, to allow for maximum efficiency in the execution of the design (McCulley and Bloebaum 1994, 1995). The efficiency of a system can be increased if certain problem-dependent parameters are minimized, such as cost, CPU time, feedbacks, or crossovers (Steward 1981; Rogers 1989). A feedback occurs when a system module requires information from another module that is located later in the design sequence. A crossover occurs when the feedbacks of two modules intersect, without any transfer of information. The development of convergence strategies for such coupled systems is of particular importance (McCulley 1996).

Another area of current research within the field of MDO is the concept of system reduction, through coupling suspension and elimination (Bloebaum 1992; Miller et al. 1995). The decomposition of a large system results in a series of smaller subsystems that are interrelated through couplings. Because of the enormity of many engineering systems, there is a need to minimize the complexities of the system, and thus the time for both design system convergence and sensitivity analysis. Thus, it is advantageous to find an analytical means for quantifying the strengths of these couplings. Couplings that are found to be weak can be suspended for a portion of the system analysis, or eliminated outright. This concept provides the foundation for system reduction strategies.

Industry is primarily interested in the way that the var-
ious participating design groups communicate. An efficient and natural way for design groups to pass information back and forth is via distributed processing. The concept of distributed processing assures that the system design tasks are computationally distributed among the participating design groups. In this way, distributed processing extends the principals of MDO to a computer network. This methodology allows for parallel communication between the design groups, and thus provides greater efficiency than a sequential computing approach. A modern day approach that is used to achieve distributed processing is the parallel virtual machine (PVM) coding language (Geist et al. 1994). PVM uses library calls and message passing to distribute tasks amongst the individual computer hosts on the network. Hence, some researchers are investigating the uses of PVM for the complex systems encountered in MDO. All of these research areas involve development of methods to increase efficiency or robustness in the MDO environment. Hence, all of these efforts require example problems on which to test their methods.

2 Motivation for creation of CASCADE

It is quite clear that the field of MDO has a great deal of potential to provide methodologies that can be used in industry. For this to happen, these MDO methodologies must be tested extensively. Researchers typically spend a great deal of time and effort developing trial systems to test their methodologies. These systems are often quite large and computationally expensive to deal with. It would, therefore, be convenient for these researchers to possess a simulator that is capable of generating, converging, and then further analysing an analytical representation of a complex engineering system. The simulator should be robust and capable of generating a system whose coupling nature is either random or user-specified. Lastly, the simulation should be realistic, in that it should allow for a distributed processing communication architecture.

This paper discusses the design and creation of an MDO-type simulator, CASCADE (Complex Application Simulator for the Creation of Analytical Design Equations). CASCADE can be used to generate coupled systems of analytical equations of user-specified size. Thereafter, CASCADE employs a system analysis to iteratively converge the generated system. To add realism to the simulation, this process can be made to take place in a distributed environment, using the PVM coding language. After the system has converged, CASCADE uses the GSE method to compute the total sensitivities of all output responses (behaviour variables), with respect to all inputs (design variables). This sensitivity information could potentially be used to analyse coupling strengths for possible suspension/elimination or could be used in an optimization sensitivity analysis. CASCADE writes each converged output (behaviour variable) equation to a separate subroutine. Researchers could potentially experiment with this sequence of subroutines to further investigate coupling strengths, sequencing issues, or convergence strategies. Lastly, for testing optimization strategies, CASCADE generates a coupled optimization problem, whose variables are both the design variables and the behaviour variables of the converged set of equations. The objective function and each constraint function of the randomly generated optimization problem are written to separate subroutines. The problem is then solved, using the CONMIN optimizer (Vanderplaats 1973).

3 Programming methodology of CASCADE

The CASCADE simulator is described by first addressing the make-up of the coupled system and the inputs required by the user, then discusses the means of achieving a converged system, and finally addresses post-convergence features.

3.1 Preliminaries

CASCADE has been designed to create unpredictable, randomly generated systems of user-prescribed size. Such systems will best represent the wide variety of MDO problems that might be of interest to researchers. For this reason, a random number generator is used to make a number of system-related decisions. For example, random number generation is used to determine both the number of terms per behaviour variable (output equation), and an initialized value for each design variable. By default, CASCADE will randomly generate the "nature" (i.e. the decision as to which subsystems are coupled to which other subsystems) of the couplings, when executed. Alternatively, the user may wish to specify the exact nature of the couplings for the system to be created. This can be accomplished by creating an input-coupling data file prior to execution of the main program. This file should contain (a) a dummy character string on the top line, (b) the integer number of subsystems in the system to be created (n) on the second line, and then (c) an (n x n) boolean matrix, whose columns dictate the coupling of the system. For further details, visit the URL listed at the end of this document.

3.2 System construction

Prior to executing CASCADE to generate and converge a system, the user must create an input file. The user assigns a variety of options in this file, including the number of subsystems, design variables, and behaviour variables in the system. (Recall that the user can alternatively specify the number of subsystems, as well as the exact coupling nature of the system by creating the coupling data file explained earlier). The user assigns numerous other options when generating the input file, including convergence options, and post-convergence options, such as GSE derivatives and a coupled optimization problem. Once these preliminaries are handled, the system can be constructed, term by term.

The terms that are generated and combined to create the coupled set of equations take the form:

\[ y_i = \sum a_j x_j^b_j + \sum c_k u_k^d_k + \cdots \]

(1)

where, \( y_i \) is the i-th output of the Y subsystem, and is a function of j design variables \( x \) and output couplings from other subsystems such as \( w_k \)'s from subsystem W. Further, each of these design variables and coupling outputs can be raised to some power \( (b_j \text{ and } d_k) \) and is premultiplied by a coefficient \( (a_j \text{ and } c_k) \).

First, the nature of every term in the system is determined, as illustrated in (1), on the first iteration of execution. For each term in the system, the sign, exponent (one