Structural Optimization 14, 129–137 © Springer-Verlag 1997

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 a 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) (Kroo 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 various 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 ($n \times n$) boolean matrix, whose columns dictate the coupling of the system. For futher 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:

$$y_i = \sum a_j x_j^{b_j} + \sum c_k w_k^{d_k} + \cdots , \tag{1}$$

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 and d_k) and is premultiplied by a coefficient (a_j 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 of a possible six choices), and coupling nature (coupling to either a design variable or to a behaviour variable) is determined. Next, the magnitudes of each term and the sensitivity of each term are determined. Design variable magnitudes are identically known as they are initialized by the random number generator, and held constant. Hence, the magnitudes of design variable-coupled terms can be identically computed on the first iteration. CASCADE generates nonhierarchical systems which require both initial guesswork of the behaviour variable magnitudes and iteration to converge. The magnitudes of the behaviour variables are not initially known. For this reason, the magnitudes of behaviour variable-coupled terms cannot be computed identically. The initial guess for the magnitude of all behaviour variables, on the first iteration, is chosen to be 1.0.

Next, upper upper and lower limits are assigned on the magnitudes of the terms in each behaviour variable. The magnitude of each term must be greater than 0.0, and less than +500.0. This assures that every term will be positive, and not excessively "large". This limitation takes place to prevent both a diverging behaviour variable, and an undefined exponentiation of a negative term. Further limited is the magnitude of the entire behaviour variable equation, once all terms in the given equation have been summed.

3.3 *Sample CASCADE system*

In this sample system, the user has decided that the system will have three subsystems. The user has also decided that subsystem 1 has two outputs "w_1" and "w_2", subsystem 2 has four outputs "y_1", "y_2", "y_3", and "y_4", and subsystem 3 has three outputs "z_1", "z_2", and "z_3". Similarly, the user has decided that each subsystem has one independent design variable (x_w, x_y, and x_z, respectively) (Fig. 1).

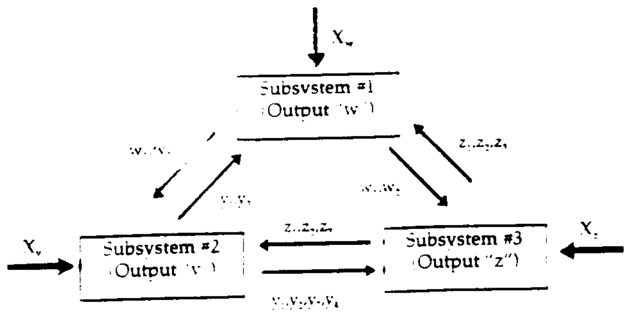

Fig. 1. System of three coupled subsystems

Recall that CASCADE will randomly determine the number of terms per output equation (ranging from 1 to 20), as well as the coupling nature, coefficient, sign, and exponent of each term in each equation. Assume that CASCADE has decided that equation w_1 (of subsystem 1) will have three terms. Table 1 summarizes CASCADE's decisions that lead to the construction of equation w_1.

As a result, equation w_1 will appear as follows:

$$w_1 = +0.967x_w^2 + 0.265y_4^{-1} - 0.087z_1^{1/3} \qquad (2)$$

CASCADE carries out a similar procedure for equation w_2 of the same subsystem, as well as equations "y" and "z" of subsystems 2 and 3, respectively. Upon creation of the system, iterative convergence procedures commence.

Table 1. CASCADE term generation for equation w_1

Equation w_1	Term 1	Term 2	Term 3
sign	positive	positive	negative
coefficient	0.967	0.265	0.087
coupling nature	des. var. (x)	beh. var. (y)	beh. var. (z)
coupling number	1	4	1
exponent	2	-1	1/3

3.4 *System convergence*

Once all equation magnitudes have been determined and found to be within an "acceptable" range, a convergence check takes place. The newly computed magnitudes of each behaviour variable, on the present iteration, are compared to the corresponding magnitudes from the previous iteration. If the differences between these values, for all behaviour variables, are less than or equal to the initial convergence criterion, then the system is determined to be converged. If not, the process repeats, and the newly computed (and more accurate) behaviour variable magnitudes are used for the next iteration. Convergence "ease" parameters have been incorporated into the program, such that the user can ease the convergence criterion by 1 and/or 2 decimal places after a user-defined number of iterations. This will take place if the system is having problems converging to the initial convergence criterion.

3.5 *Extension to PVM*

The above procedure is an example of what occurs for a single computer operation. This procedure is slightly modified to operate in a distributed computing environment, via PVM. A master machine enrolls in PVM, and performs the preliminaries: namely, random number generation, and reading in the input file. The master then spawns slave tasks on the other hosts that were detected on the virtual machine that is presently available. The master then packs and sends array data for one subsystem, to each slave. Each slave receives and unpacks the data sent to it by the master, and then performs the analysis described in Section 3.2, for the one subsystem that was sent to it. Each slave then sends the newly computed data back to the master for a convergence check. The process repeats until the system converges. Post-convergence features take place on the master machine, and the slaves exit their processes (Fig. 2).

3.6 *Post-convergence features*

Once the system has converged, CASCADE performs a variety of post-convergence features, that could be greatly beneficial to an MDO-researcher. Character strings are used to write every converged behaviour variable of the created system to a separate subroutine. These subroutines could be used in a system reduction analysis, to view the effects of the suspension and/or elimination of relatively weak couplings. Alternatively, each converged subroutine could be optimally sequenced, and then re-converged, to test sequencing or convergence strategies. Also, these subroutines can be used to test distributed computing approaches.

The GSE approach is used to attain the total derivative matrix of the system that has been created. Equation (3) lists the GSE for a system with two subsystems, A and B. In this case, each subsystem has an independent input, "X", and

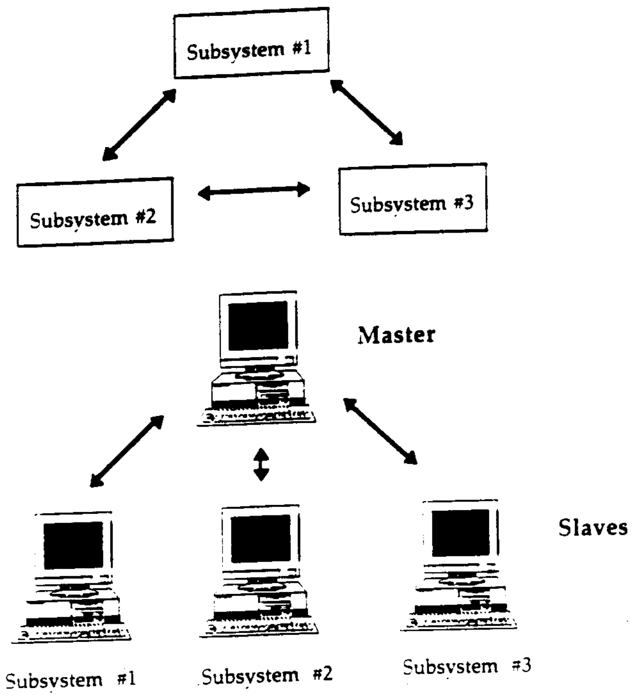

Fig. 2. Extension of system analysis to PVM

a dependent input "Y", which is the output from the other subsystem. The left- and right-hand side partial derivative matrices seen in (3) can be computed at the subsystem level, as the sensitivity of each output equation of the system can be computed, with respect to both (a) every other behaviour variable and (b) every subsystem design variable.

$$\begin{bmatrix} 1 & -\frac{\partial Y_A}{\partial Y_B} \\ -\frac{\partial Y_B}{\partial Y_A} & 1 \end{bmatrix} \begin{bmatrix} \frac{dY_A}{dX_A} \\ \frac{dY_B}{dX_A} \end{bmatrix} = \begin{bmatrix} \frac{\partial Y_A}{\partial X_A} \\ 0 \end{bmatrix}$$
$$\begin{bmatrix} 1 & -\frac{\partial Y_A}{\partial Y_B} \\ -\frac{\partial Y_B}{\partial Y_A} & 1 \end{bmatrix} \begin{bmatrix} \frac{dY_A}{dX_B} \\ \frac{dY_B}{dX_B} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\partial Y_B}{\partial X_B} \end{bmatrix} \quad (3)$$

These matrices are then normalized, and an LU-decomposition is used to attain the normalized total derivative matrix. The total derivatives are recovered by "unnormalizing" each element of this normalized matrix. This matrix can be used to assess the coupling relations that are weak relative to the others. It is possible that these weak couplings be eliminated from the system analysis, or at least suspended for part of it. These derivatives could also be used for some form of formal system optimization procedure.

A statistical output listing of the constructed system is provided. The file lists CPU times, and the number of iterations required to converge the system. The file also generates a term-by-term listing of the entire system that has been generated. (Note. By using the same input seed value, repeat systems can be generated. This will be useful for method comparison studies.)

Finally, the user has the option of generating an optimization problem, whose variables are the coupled variables of the converged CASCADE system. These variables include design variables, which were initialized and held constant for the convergence procedure, and behaviour variables, which were initialized and iteratively updated and eventually converged. For the purpose of the optimization problem, the behaviour variables are treated as constants, and the design variables are altered so as to provide the global optimum solution. The traditional optimization problem statement is stated as follows:

minimize $F(\mathbf{X})$,

subject to

$$g_j(\mathbf{X}) \leq 0, \quad j = 1, m, \quad h_k(\mathbf{X}) = 0, \quad k = 1, n,$$
$$\mathbf{X}_i^\ell \leq \mathbf{X}_i \leq \mathbf{X}_i^u, \quad i = 1, n. \quad (4)$$

F is known as the objective function, and is the function being minimized, g_j are the inequality constraints, h_k are the equality constraints, and $\mathbf{X}$ is the vector of design variables. In the analysis of a complex engineering system, one may need to minimize a function whose variables are coupled, and whose constraints are also functions of these coupled variables. Such a problem statement may be stated as follows:

minimize $F(\mathbf{X}, w, y, z)$,

subject to

$$g_1(\mathbf{X}, w, y, z) \leq 0, \quad g_2(\mathbf{X}, w, y, z) \leq 0,$$
$$g_3(\mathbf{X}, w, y, z) \leq 0, \quad \mathbf{X}_i^\ell \leq \mathbf{X} \leq \mathbf{X}_i^u, \quad i = 1, n. \quad (5)$$

Notice that the objective function and inequality constraint functions are functions of both the design vector $\mathbf{X}$, and of the coupled behaviour variables w, y, and z. Because CASCADE will randomly generate the nature (namely, the coefficient, exponent, and coupling) of each term of the objective and constraint functions, provisions must be made to guarantee that an optimum solution will exist for the problem that is generated. First of all, the number of inequality constraints must be less than or equal to the number of design variables, to assure that the design space will not be overconstrained. The computer simulator development of Padula *et al.* (1986, 1987) demonstrated that each constraint function must be one of three forms: (a) strictly decreasing, (b) strictly increasing, or (c) asymptotically increasing; hence the three constraints in (5). Together, these three types of equations can simulate the behaviour of most engineering systems. A sample optimization problem, as generated by CASCADE, is seen in the following equation:

minimize OBJ =

$$f_1(x_{11})^2 + f_2(x_3^2)^{-1} - f_3(w_1)^{1/3} + f_4(z_3)^{1/2},$$

subject to

$$g_1 = -20r_1 + a_1/(x_{11}) + a_2/(y_3) + b_1/(x_{11}^* w_2),$$
$$g_2 = -20r_2 + a_3(x_{32}) + a_4(y_1) + b_2(z_2^* x_{22}),$$
$$g_3 = -20r_3 + a_5(z_2)^{0.5} + a_6(x_{33})^{0.5} + b_3(z_2^* x_{13})^{0.5}. \quad (6)$$

Refer to the nomenclature of Fig. 1. In this example, the a's, b's, r's, and f's are randomly generated constant coefficients. Again, the x's are elements of the design vector $\mathbf{X}$, and the w's, y's, and z's are coupled behaviour variables from the original set of equations generated by CASCADE. The exponents and couplings in the constraint equations are randomly generated. The signs of the objective function terms are randomly generated, as well. The first term in each constraint equation is set to be a negative constant. All other terms in each constraint equation are positive couplings, which, when summed with the negative first term, must remain less than or equal to 0 for constraint satisfaction.

To prevent an unbounded optimization problem, side constraints are generated for each design variable. Lower bounds are set to be a random number between 0 and 1. The lower bound is set to be greater than zero, so as to prevent undefined exponentiations of fractional powers. The upper bound is set to be a random number between 0 and 150.0, and larger than the lower bound. Recall that each design variable is initialized and held constant during the convergence process of the system of coupled equations. That initialized value is used as the starting value for the optimization problem. CASCADE assures that the side constraint region for each design variable bounds this starting value.

Once properly generated, the optimization problem is solved by using Vanderplaats' CONMIN program, which is a FORTRAN program for constrained function minimization. CONMIN makes use of the method of usable-feasible directions. CONMIN has been set to allow for user-supplied gradients, which are more accurate and less costly than internally computed finite-difference gradients. Because the objective and constraints are known functions of the design and behaviour variable vectors, their gradients are computed precisely by analytical means, with relative ease. Note that the total derivative of the objective function or a constraint function with respect to a design variable will be a function of the partial derivative of the function with respect to the design variable, plus a partial derivative component from each output equation. Equation (7) lists the derivatives of an arbitrary objective function and constraint function (i, j) with respect to design variable w from Fig. 1.

$$\frac{\mathrm{d}F}{\mathrm{d}X_w} = \frac{\partial F}{\partial X_w} + \frac{\partial F}{\partial w}\frac{\mathrm{d}w}{\mathrm{d}X_w} + \frac{\partial F}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}X_w} + \frac{\partial F}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}X_w}$$

$$\frac{\mathrm{d}g_{i,j}}{\mathrm{d}X_w} = \frac{\partial g_{i,j}}{\partial X_w} + \frac{\partial g_{i,j}}{\partial X_w}\frac{\mathrm{d}w}{\mathrm{d}X_w} + \frac{\partial g_{i,j}}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}X_w} + \frac{\partial g_{i,j}}{\partial z}\frac{\mathrm{d}z}{\mathrm{d}X_w}. \quad (7)$$

Note that the total derivative terms on the right-hand sides of the above equations come from the solution of the GSE.

After the optimization problem has been generated and solved, the objective function and constraint equations are each written to a separate subroutine, by using character strings. The user has the option of generating either FORTRAN or ANSI C subroutines. The initial design variable magnitudes and behaviour variable magnitudes are written to a subroutine, as well.

4 Outputs of CASCADE for varying system sizes

Because the thrust of this work focuses on the development of the CASCADE simulator, it is difficult to discuss results in the traditional sense. The worth of this simulator can only be measured in its use by the MDO community. It should be noted, however, that CASCADE has already been used extensively for optimal sequencing and convergence strategy studies, as well as for system reduction and for MDO method testing. In particular, researchers from the MODEL laboratory at SUNY Buffalo have used the simulator in various capacities. The simulator has been used to generate test systems for developing sequencing strategies for time and cost considerations (McCulley and Bloebaum 1996), for the evaluation of weak couplings for suspension/elimination (English et al. 1996), and for using Java as a distributed computing tool (Becker and Bloebaum 1996). Likewise, the simulator has been used to generate test systems to examine the extension of the CSSO method to the analysis of mixed-variable coupled systems (Chi and Bloebaum, 1996). In this paper, however, the potentials of distributed computing using PVM are explored and discussed in some detail, with resulting computational times presented

4.1 *Clock time normalized per iteration*

The clock time required to build and converge a coupled system is a primary concern of a system analyst. CASCADE has been used here to generate a wide variety of systems and to analyse the execution times involved. This was done in an effort to better understand the potential advantages, as well as the drawbacks of using distributed computing in an MDO-setting. All execution times have been normalized by the number of iterations that were required to converge the system to the specified criterion of 1.0E-4. The first result is seen in Fig. 3, which plots the "single computer" results. The computers that were used are called Moriarty and EAS00; the former is a 12 processor mini-supercomputer, and the latter is a Sun-Sparc 4 workstation.

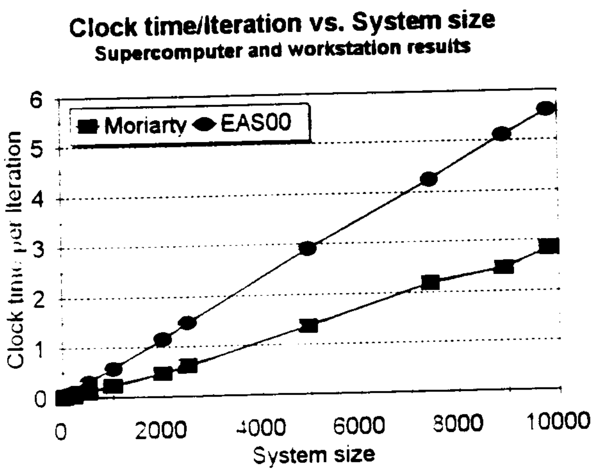

Fig. 3. Clock time results: single computer

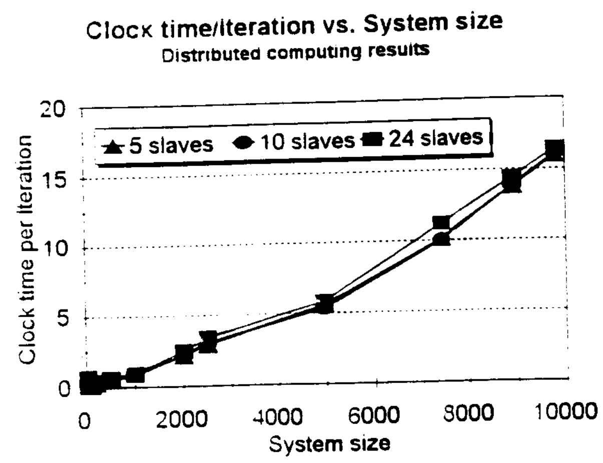

Fig. 4. Clock time results: distributed computing

As expected, the supercomputer substantially outperforms the workstation. The supercomputer normalized clock times are at least twice as fast, for most system sizes. Most systems require from 40-60 iterations to converge, so the absolute clock times for the supercomputer are on the order of 180 seconds, for a 10000 behaviour variable system. Moreover, there is a near linear correlation between normalized clock time and system size, for both computer scenarios.

From several points of view, the distributed computing data of Fig. 4 does not live up to expectations. Here, three scenarios are investigated, with 5, 10, and 24 slave machines. Both the master and slave machines are Sun Sparc 4 workstations, similar in nature to EAS00. As expected, there is a general upward trend in normalized clock time, with an increase in system size, for all three scenarios (5, 10, and 24 slave machines). However, the clock times are larger than those corresponding to the single machine data of Fig. 3. At first glance, this appears to be counter-intuitive. One would think that a system that is solved by multiple machines would be constructed and converged more quickly than a system that is analysed on one machine. This was not found to be the case.

Upon comparing the three curves themselves in Fig. 4, one sees that for the most part, systems of all sizes required roughly the same amount of time to converge, regardless of the number of slave machines being used. This again appears to be counter-intuitive. It would seem likely that a system solved using distributed computing techniques would converge faster, when using a *greater* amount of slave computers on the virtual machine. This was not found to be the case. Clearly, the question then becomes - "why?".

As highly coupled as the system of equations that CASCADE generates are, the systems are not complicated enough to fully exploit the strengths of distributed computing. Message passing dominates the clock time involved with converging a system when using distributed computing techniques. To explore this situation further, "sleep" times were introduced into each subsystem analysis. In other words, each subsystem analysis is performed (either in a sequential or a distributed computing environment), and then the program execution pauses for a user-specified number of seconds. This sleep time is incorporated in order to simulate a computation with a higher level of complexity, hence requiring a longer length of time for analysis.

4.2 *Effects of sleep time on each subsystem iteration*

Sleep times were initially experimented with on the Moriarty supercomputer, the EAS00 workstation, and the 5 and 10 slave distributed computing scenarios.

Figure 5 is a plot of clock time vs. system size for all four scenarios, with a 0.1 sleep time. With this amount of sleep time, the distributed computing scenarios clearly overtake both Moriarty and EAS00, for the full range of system sizes. However, there is still only minimal differentiation between the 5 and 10 slave scenarios. To see if any numerical sleep time would have an impact on the variation between distributed computing results with varying numbers of slave machines, the sleep time was further increased to 0.5 seconds. The results are seen in Fig. 6. Here, the 5 and 10 slave machine curves are split apart, with the 10 slave scenario clearly outperforming the 5 slave scenario. This is the intuitive result that was hoped for all along.

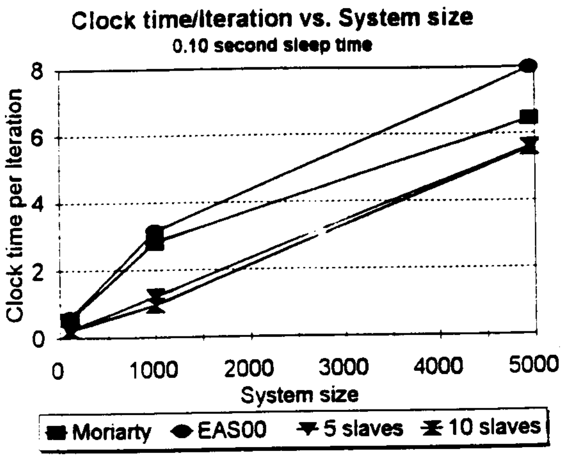

Fig. 5. Normalized clock time - 0.10 sec sleep time

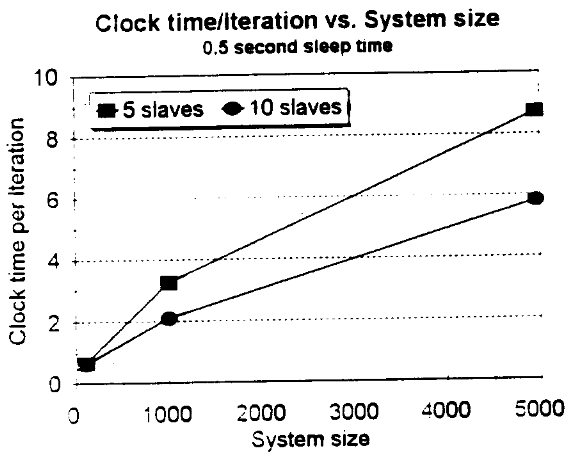

Fig. 6. Normalized clock time - 0.50 sec sleep time

Hence, each subsystem analysis of CASCADE-generated systems required roughly one half of one second longer so that the benefits of distributed computing could be fully demonstrated. This is an extremely important point for those who wish to use CASCADE to simulate a distributed computing environment. It is necessary to have sufficiently large calculation times so that message passing does not dominate and skew results. Of additional interest is the calculation of system derivatives, which can be used in a variety of MDO applications.

4.3 *Effects of normalization on GSE solution*

Table 2 lists condition number vs. system size, for both normalized and not-normalized scenarios. The condition number provides a general indication as to how well-conditioned the partial derivative matrices were, when used to attain the total derivative matrix. Thus, the condition number reflects the reliability of the numerical estimate that was attained for the total derivative matrix. (A *low* condition number is desirable.) The normalized data has matrix condition numbers that are far lower than those for the not-normalized data. Using not-normalized data, a total derivative solution *could not* *be found* for the 990 equation system example, as depicted in the table. For this system size, the condition number is infinite, for all intents.

Table 2. Condition number comparison of normalized/non-normalized data

Number of system equations	Condition number: normalized sensitivities	Condition number: not-normalized sensitivities
100	40.06	60634.
300	37.10	101310.
500	390.70	101182
700	60.09	58305.
990	345.83	47565084.*

* no solution attained

Figure 7 is a plot of the clock time required to compute the total sensitivity matrix vs. the size of the system, for both the normalized and not-normalized scenarios. For smaller systems, the clock times for both scenarios are comparable. As the system size increases, the clock time required to compute the sensitivity matrix is slightly shorter for the normalized matrices than for the not-normalized matrices. This reduction in clock time is likely attributable to the beneficial numerical conditioning provided by the matrix normalization procedure. In other words, normalization techniques not only result in a more reliable numerical estimate of the total derivative matrix, but they do so in a slightly shorter period of time.

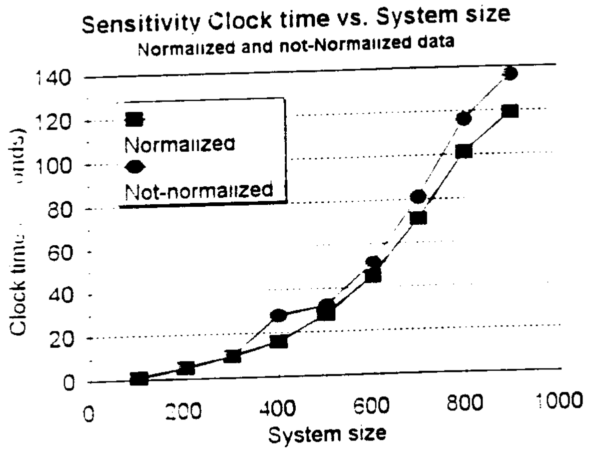

Fig. 7. Sensitivity CPU times vs. system size

There does not seem to be any correlation between condition number and system size. A large condition number will result when the degree of coupling is high. A system whose coupling nature is generated randomly by the user can possess a wide range of coupling complexity, regardless of the size of the system. In other words, highly coupled systems (and hence large condition numbers) can result for small or for large systems.

4.4 CONMIN results of coupled optimization problems

Numerous coupled systems and corresponding optimization problems have been created with the CASCADE simulator. Because it is difficult to verify that the solution produced by CONMIN is in fact the global optimum solution, the results for small system sizes have been compared to those obtained using 10^8 iterations of an exhaustive search algorithm. This comparison can be viewed in Table 3.

Table 3. Comparison of optimization results for CONMIN and exhaustive search

# subsystems (# constraints)	Initial feasible solution	CONMIN optimum	Exhaustive-search optimum
3 (6)	-693.0	-764.2	-744.9
3 (6)	-504.9	-561.3	-548.1
3 (6)	+227.3	-97.8	-85.0
5 (10)	-135.6	-1916.5	-1800.3

It is clear from this table that the optimum solution attained with CONMIN is superior to that attained with the exhaustive search, for all four test cases. If CONMIN were not consistently producing the global or near-global optimum solution, it would seem likely that the exhaustive search algorithm would attain a superior solution, for at least one of the above scenarios. This was not found to be the case, thus providing greater confidence in the CONMIN-generated results.

5 Summary and discussion

The motivation for MDO is to reduce the time and cost required for the design process. Most coupled systems are nonhierarchical in nature, and require an iterative scheme (and initial approximations) to converge. Task sequencing researchers attempt to find the optimal sequence (order) in which to analyse the system modules, to gain the converged solution in the least amount of time. System reduction researchers seek to effectively reduce the size of a coupled system, with a minimal loss in accuracy. Researchers have attempted to temporarily suspend and/or permanently eliminate output couplings that were comparatively found to be less substantial.

Before such MDO-strategies can be implemented in the design process of such large systems as automobiles and aircraft, they must be tested. A method for analytically simulating real-life, large system couplings is necessary. To this end, the authors have designed a computer program, CASCADE (Complex Application Simulator for the Creation of Analytical Design Equations). CASCADE accepts user inputs to randomly construct and then converge a large system of coupled equations. This system of equations should be viewed as an analytical representation to a real-life design scenario. Construction and convergence was first implemented using a single computer format, cycling though all of the subsystems in the system, one-by-one, in a sequential manner. The procedure was later modified for solution in a distributed computing environment, using Parallel Virtual Machine (PVM).

Once the system is constructed and converged, CASCADE offers numerous post-convergence features. The global sensitivity equation (GSE) method can be used to compute the total derivatives of the system outputs, with respect to the inputs. This is accomplished by first computing sensitivities on a local level. A second important feature is the option to write each equation of the converged system to a separate subroutine. This could be beneficial to sequencing researchers, who might perturb the design variables of the converged system. and then analyse the various ordering possibilities of the subroutines to see which sequence would attain a new converged solution most quickly. A third important feature is the option to write the converged system to a parameters file. This file provides a comprehensive listing of the nature of each term in the system that has been constructed. as well as other system statistics. A final important feature is the option to create an optimization problem, whose objective function and constraints (inequality and side) are functions of the design and behaviour variables of the just-converged system of equations. The problem is solved by using the GSE derivatives and the CONMIN optimizer. The generated objective function, constraint functions. and input variables are subsequently written to subroutines. by using character strings.

The previous section analysed the time-related results of numerous executions of CASCADE. The single computer results saw an increase in CPU time with an increase in system size. As expected. systems that were solved using the Moriarty supercomputer solved much faster than those solved on the local EAS00 workstation. Unfortunately. the unaltered PVM results did not live up to expectations. The CPU times for the parallel machine-generated systems were larger than those for the single computer systems. Moreover. CPU times per iteration were invariant. when a larger number of slave computers were used on the virtual machine. It is probable that the time required to pass information from the master machine to the slave machines is what dominated the convergence time for the PVM scenarios.

To reduce the impact of message passing, sleep times were introduced into the convergence procedure. to simulate calculations of higher complexity. On both the single computer and distributed computing scenarios. each subsystem analysis was performed. and then followed by a user-specified period of sleep. With 0.5 seconds of sleep time. the advantages of using a larger number of slave machines for distributed computing become evident. The bottom line is that the CASCADE-generated systems alone are not complicated enough to exploit the benefits of distributed processing.

Matrix normalization was found to be beneficial. for the computation of the total derivative matrix by using the GSE method. The condition number of the solution matrix was lower when normalization techniques were employed: indicating a more reliable numerical estimate. The CPU times required to attain the normalized derivative matrix were also lower. than those required to attain not-normalized derivative matrices.

The results of the solution to the coupled optimization problem were compared to those obtained from an exhaustive search algorithm. for smaller problem sizes. In all cases. the CONMIN solution is superior to that of the exhaustive algorithm. This comparison provides a certain degree of confidence that CONMIN is providing the global optimum solution the majority of the time. if not always.

6 Future work

One step that can be taken for the advancement of this research would be to improve the results associated with distributed computing and PVM. A method must be found to more effectively pass the large array data from the master to the slave. Message passing was found to be the dominating factor in the time taken to build and converge the systems, while using distributed computing techniques. An interface that could enable the PVM message passing to bypass select network layers, without performance degradation, and reduce message passing time would be beneficial. Eventually a conversion to an ATM-based network is foreseen (Lakshminarayan 1993). Further time-related improvement could be made by setting up a virtual machine whose hosts consist of the numerous processors on a multiprocessor supercomputer.

The size of systems created by CASCADE has been limited by the static dimensioning allowances of the FORTRAN programming language. If this were overcome, systems of infinite size could potentially be constructed and converged. This would prove that the principals of MDO could be extended to systems of any imaginable size. with couplings having any imaginable complexity.

A final addition is presently being added to CASCADE - a user-friendly Java-based graphical user interface. Because the interface is coded in Java. any computer architecture can potentially make use of it. The interface will allow the user to easily assign all input options required to construct and converge a coupled system of equations. The interface will also allow the user to choose the desired post-convergence features. such as creation of the total derivatives via GSE. construction of an optimization problem. and so forth. Once all options are assigned. the user will press a button which will simultaneously create the input file that is required to execute CASCADE. and then implement the system command to physically execute CASCADE. All pertinent output files will be written to the user's directory upon execution.

To execute CASCADE and make use of this interface. the user will need a Java interpreter. as well as the compiled CASCADE code (written in FORTRAN). The necessary files will be available in early 1997 via the MDO test suite (Padula *et al.* 1996). provided by the NASA Langley Research Center. whose URL is as follows:

http://fmad-www.larc.nasa.gov/mdob/mdo.test/class1prob3.html

References

Becker. J.; Bloebaum. C.L. 1996: Distributed computing for multidisciplinary optimization using Java. Sixth AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization (held in Bellevue. WA)

Bloebaum. C.L. 1989: Global sensitivity analysis in control-augmented structural synthesis. *AIAA Student J.*

Bloebaum. C.L. 1991: *Formal and heuristic system decomposition methods in multidisciplinary synthesis.* Ph.D. Thesis. University of Florida. Gainsville. FL

Bloebaum. C.L. 1992: An intelligent decomposition approach for coupled engineering systems. Fourth AIAA/USAF/NASA/OAI Symp. on Multidisciplinary Analysis and Optimization (held in Cleveland. OH)

Chi. H.-W.; Bloebaum. C.L. 1996: Concurrent subspace optimization for mixed-variable coupled engineering systems. Sixth AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization (held in Bellevue. WA)

English, K.; Miller, E.; Bloebaum, C.L. 1996: Total derivative based coupling suspension for system reduction in complex design. Sixth AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization (held in Bellevue, WA)

Geist, A.; Begueiin, A.; Dongerra, J.; Weicheng, J.; Mancheck, R.; Sunderam, V. 1994: *PVM: Parallel virtual machine - A user's guide and tutorial for networked parallel computing.* Cambridge, MASS: The MIT Press

Hajela, P.; Bloebaum, C.L.; Sobieszczanski-Sobieski, J. 1990: Application of global sensitivity equations in multidisciplinary aircraft synthesis. *J. Aircraft* **27**, 1002–1010

Hulme, K.F.; Bloebaum, C.L. 1996: *Development of CASCADE - A multidisciplinary design optimization test simulator for use in distributed computing environments.* Masters Thesis, University of Buffalo, Buffalo, NY

Kroo, I. 1995: Decomposition and collaborative optimization for large scale aerospace design. ICASE/LaRC/SIAM Workshop on Multidisciplinary Optimization (held in Hampton, VA)

Lakshminarayan, K. 1993: *ATM networking and multimedia - A white paper.* Sun Microsystems Computer Corporation, Revision X

McCulley, C.; Bloebaum, C.L. 1994: Optimal sequencing for complex engineering systems using genetic algorithms. Fifth AIAA/USAF/NASA/OAI Symp. on Multidisciplinary Analysis and Optimization (held in Panama City, FL)

McCulley, C.M.; Bloebaum, C.L. 1995: *A genetic tool for optimally sequencing the design of complex engineering systems.* Masters Thesis, University of Buffalo, Buffalo, NY

McCulley, C.; Bloebaum, C.L. 1996: Complex system design task sequencing for cost and time considerations. Sixth AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization (held in Bellevue, WA)

Miller, E.; Bloebaum, C.L. 1995: *Coupling suspension and elimination in multidisciplinary design optimization.* Masters Thesis, University of Buffalo, Buffalo, NY

Padula, S.L.; Sobieszczanski-Sobieski, J. 1987: A computer simulator for development of engineering system design methodologies. *NASA Technical Memorandum*

Padula, S.L.; Young, K.C. 1986: Simulator for multilevel optimization research. *NASA Technical Memorandum*

Padula, S.L.; Alexandrov, N.; Green, L.L. 1996: MDO test suite at NASA Langley Research Center. Sixth AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization (held in Bellevue, WA)

Rogers, J.L. 1989: DeMAID – A design manager's aide for intelligent decomposition: user's guide. *NASA Technical Memorandum 101575*

Sobieszczanski-Sobieski, J. 1982: A linear decomposition method for optimization problems - Blueprint for development. *NASA Technical Memorandum 83248*

Sobieszczanski-Sobieski, J. 1990: Sensitivity of complex, internally coupled systems. *AIAA J.* **28**, 153-160

Steward, D.V. 1981: *System analysis and management.* New York: Petrocelli Books

Vanderplaats, G.N. 1973: CONMIN - A FORTRAN program for constrained function minimization. *NASA Technical Memorandum*

Received March 1[illegible], 1997
Revised manuscript received June 1[illegible], 1997