Efficient Shape Optimization of Crashworthy Structures using a New Substructuring Method

Ronald C. Averill

Red Cedar Technology
4572 S. Hagadorn Rd., Suite 1-E
East Lansing, MI 48823
r.averill@redcedartech.com

Abbreviations:
HEEDS – Hierarchical Evolutionary Engineering Design System
COMPOSE – COMPonent Optimization within a System Environment

Keywords:
crashworthiness, optimization, shape, decomposition, HEEDS, COMPOSE

ABSTRACT

A new methodology is introduced to drastically reduce the time and effort required to perform vehicle crashworthy design optimization on components or subsystems whose performance is strongly coupled to that of the complete system to which they belong. The current approach, named COMPOSE, reduces the CPU time for such design studies by a factor of 10^1 – 10^3, depending upon the problem definition. The main advantage of COMPOSE is that potentially thousands of design iterations can be performed in about the same amount of time required to perform a few (4-8) system level evaluations. With this approach, engineers can perform design studies and thoroughly explore new design concepts that were previously ignored due to their computational expense. This technology has been implemented into a software code called HEEDS (Hierarchical Evolutionary Engineering Design System). Using LS-DYNA explicit as the finite element solver within the HEEDS optimization environment, this process has been applied to several automotive lower compartment rail designs, resulting in significant gains in performance compared to baseline rails designed by experienced engineers. An example application of this method is described herein.

INTRODUCTION

The design of structures is often driven by many competing criteria such as lower cost, weight reduction, enhanced safety and multi-disciplinary performance, and manufacturability. In addition, the introduction of new manufacturing processes and materials significantly increases the available design space, or the set of all possible designs for a problem. In order to explore this large design space more effectively while trying to reduce design cycle times, engineers can now take advantage of automated design optimization technology. These tools can greatly decrease the time required to identify a set of feasible, or even near-optimal, designs prior to building and testing the first prototype. Moreover, these tools can also provide designers with the freedom and power to seek creative solutions that are not obvious to even the most experienced engineer. This is true in general but particularly so with shape optimization problems, which can involve potentially hundreds of design variables. Moreover, in the shape design of crashworthy systems, the high degree of nonlinearity in the system response makes it difficult to rely solely on intuition to estimate the effects of simultaneously modifying numerous design parameters.

In large complex engineered systems, often only a subsystem or a small part of the system design needs to be modified to adapt or improve performance in some way. For example, to improve frontal crash safety in an automobile, an engineer might focus design changes on only the vehicle lower compartment rails and the bumper. Or, a roll-over crash design study may focus on only a roof rail.

In most cases of design for crashworthiness the subsystem behavior is strongly coupled to that of the overall system in such a way that even small changes to the subsystem can strongly affect the *interactions* between the system and subsystem. In these cases, design optimization of the subsystem usually requires that a mathematical/computational model of the complete system be used so that these interactions can be taken into account directly. These full system level models are often very large and complicated, and thus a significant amount of CPU time (e.g., 10-30 hours) is required to simulate the performance of each new design scenario. Because many (100-1,000) potential design evaluations might be necessary to perform a high fidelity design optimization involving 10-250 design variables, it would take many weeks to perform a design optimization study on even a small subsystem.

A new methodology is introduced here to drastically reduce the time and effort required to perform design optimization on subsystems whose performance is strongly coupled to that of the complete system to which they belong. The current approach reduces the CPU time for such design studies by a factor of $10^1 – 10^3$, depending upon the problem definition. Further, this approach has the potential to yield better results than is possible by directly applying most optimization methods to a full system model because a much more thorough search is performed, and robustness of the design can be enforced within the current approach. This new technology is called COMPOSE, which stands for COMPonent Optimization within a System Environment. With this approach, engineers can achieve designs with significantly higher performance and robustness, and in much less time.

BACKGROUND

Optimization is a process that seeks to identify the best design according to a specified measure, or objective function, while also satisfying certain restrictions, or constraints. But mathematical methods used for optimization may not always identify the best design, and in these cases we seek to find as good a design as possible, or at least a design that is better than the existing one. Herein, use of the word optimal is intended to mean best, or as good as possible, or better than before, depending upon what is possible in that context and application. In practical situations, we often seek a design that is optimal with respect to robustness and reliability rather than strict performance under a specified set of deterministic environmental circumstances.

Gradient-based optimization techniques have been applied successfully to many optimization problems (e.g. [6-9]). However, these methods have several drawbacks. First, they tend to find quickly and get stuck on local extrema [9]. In addition, gradient methods are not suitable for finding singular extrema, or for optimizing problems with discontinuous or noisy response functions (e.g., crash problems).

Optimization methods based on genetic algorithms (GAs) and other evolutionary methods have recently been applied to various engineering problems [1-6,10-18], and have demonstrated the potential to overcome many of the problems associated with gradient-based methods. However, the need of GAs to evaluate many alternative designs often limits their application to problems in which the design space can be made sufficiently small, even though GAs are most effective (relatively) when the design space is large.

The concept of multi-level solution of design optimization problems has been investigated for more than 20 years (see [19-23]). Yet its applications have often been limited by the specific nature of the algorithm, or by the requirement to calculate sensitivity derivatives of the subsystem parameters with respect to the system level response. In the current study, the shortcomings of existing multi-level decomposition methods are overcome in a general way, so that a generalized robust method for solving several very wide classes of multi-level design optimization problems can be developed.

We are concerned here with large optimization problems, where large may refer to the number of design variables and/or the CPU time required to evaluate the objective function for a particular design candidate. In such cases it is common to break the problem into smaller parts, or *subsystems*, using *decomposition*.

Decomposition may be applied to the optimization problem itself, to the physical/temporal domain of the system, and/or to the disciplinary responses in a multidisciplinary problem. The present discussion focuses primarily on spatial decomposition, wherein the physical system is decomposed into several subsystems. The COMPOSE algorithm is not limited to such problems, however.

After an optimization problem is decomposed, the solution procedure may take one of several forms. Among the more popular methods is a multi-level optimization procedure. For example, in a two-level optimization procedure the optimization of the subsystem variables,

$\mathbf{x}^{\mathbf{i}}$, is nested inside an upper-level optimization of the global variables, **z**. It is also possible to define a third set of variables, **y**, that are output from one subsystem and input to another subsystem [23]. An iterative approach can then be used to coordinate the identification of the subsystem and global variables that jointly optimize the system. Most such iterative approaches depend upon the calculation of sensitivity derivatives of the optima of each subsystem with respect to changes in the global variables, **z**. Often, the calculation of these derivatives is either very difficult or computationally very expensive. In some cases, the sensitivity derivatives are discontinuous. The cost of calculating the sensitivity derivatives depends in part on the front of interaction between the subsystems and the number of design variables.

A solution procedure that does not require the calculation of sensitivity derivatives would be beneficial in many applications. Such an approach is often called direct iteration, or *fixed-point iteration*. This technique, however, has less than desirable convergence characteristics when applied to some classes of problems. Namely, problems in which large changes in the interaction variables occur during the iteration process may not converge to a near optimal solution, and may fail to converge at all.

Here, several modifications to the fixed-point iteration algorithm are discussed that significantly enhance its ability to convergence when applied to multilevel optimization of large problems.

GENERAL PROBLEM STATEMENT

Consider any continuous or discrete system that exists in the domain Ω, as shown in Figure 1(a). The spatial and temporal performance of the system under a prescribed set of environmental conditions (generalized loads) can be described mathematically by equations (e.g., differential, integral, algebraic, etc.) in terms of primary variable(s) denoted *u(x,y,z,t)* and the secondary variable(s) denoted *f(x,y,z,t)*. The boundary of Ω is denoted as Γ.

Within the system domain Ω, one or more subsystems $\Omega_{subsystem(i)}$ (*i=1* to *N*) may be identified, as shown in Figure 1(b). The only restriction on the definition of these subsystems is that their domains may not overlap. Subsystems *i* and *j* must not have any common interior points for all *i,j=1* to *N*, but subsystems may have common boundary points. The subsystem boundary $\Gamma_{subsystem(i)}$ represents the boundary between subsystem *i* and the remainder of the system, as shown in Figure 1(c). We assume here without loss of generality that:

$$u = \hat{u}_{subsystem(i)} \qquad \text{on } \Gamma_{subsystem(i)} \tag{1}$$

Of interest here is the common situation in which the performance of one or more subsystem designs is to be optimized by changing one or more characteristics (design variables) of the subsystem(s). The subsystems do not share any design variables, and the remainder of the system is fixed so there are no global design variables. In this context, a subsystem is optimized when a specified objective function is minimized or maximized, including the special case in which the subsystem satisfies a particular performance target. The subsystem designs may also be subject to a set of constraints that must be satisfied. The optimization is performed by finding the simultaneous values of a set of design variables that extremize the objective function while satisfying all constraints. Mathematically, the optimization statement within each subsystem may take the form:

Minimize (or maximize):	$F_i(x_1,x_2,\dots,x_n)^i$
such that:	$G_{ij}(x_1,x_2,\dots,x_n)^i < 0, \quad j=1,2,\dots,p_i$
	$H_{ij}(x_1,x_2,\dots,x_n)^i = 0, \quad j=1,2,\dots,q_i$

(2)

where $(x_1,x_2,...,x_n)^i$ are design variables in subsystem *i*
$F_i(x_1,x_2,...,x_n)^I$ is the objective (performance) function in subsystem *i*
$G_{ij}(x_1,x_2,...,x_n)^i$ are inequality constraints in subsystem *i*
$H_{ij}(x_1,x_2,...,x_n)^i$ are equality constraints in subsystem *i*

The problem statement in Equation (2) is intended to include optimization problem statements in the broadest sense, including multi-objective optimization.

In some cases, even major changes to a subsystem design do not strongly affect the *interactions* between the system and the subsystem(s). In other words, there are some systems in which the subsystem essential boundary conditions $\hat{u}_{subsystem(i)}$ experience small or no change when the values of design variables in any subsystem are modified. In these cases, the subsystem *i* can be redesigned in isolation using mathematical models involving only the domain $\Omega_{subsystem(i)}$, which should be smaller and simpler than that of the entire system. The system contributions are included through the boundary conditions $\hat{u}_{subsystem(i)}$.

On the other hand, there are many cases in which the subsystem behavior is strongly coupled to that of the overall system in such a way that even moderate changes to a subsystem can strongly affect the *interactions* between the system and subsystem(s). This category of problem is the focus of the present study. The objective of the present work is to drastically reduce the time and effort required to perform design optimization on subsystems whose performance is strongly coupled to that of the complete system to which they belong.

Let us assume that a given design optimization statement as in Equation (2) requires that a minimum number of design evaluations be performed, this number of evaluations depending primarily upon the number of design variables, the nature of the design space, and the optimization search algorithm employed. Then, a reduction in the computational effort required to optimize a subsystem must be achieved by reducing the computational effort to evaluate each design scenario. Here, a technique is sought in which most design evaluations can be performed using the subsystem mathematical models, which should be much smaller and computationally more efficient than the complete system level model. But such an approach must also account for the sometimes strong interactions between the performance of the system and the subsystem(s).

DESCRIPTION OF COMPOSE

In a typical design optimization problem, the goal is to design a system so that it behaves in a prescribed or optimal manner in a given environment or under a set of prescribed conditions. The challenge of the current problem is to simultaneously identify *both* a subsystem that is optimal according to a specified criterion *and* the subsystem boundary conditions under which the subsystem should behave optimally.

In the general case, the subsystem boundary conditions associated with the optimal design cannot be known until the design approaches its optimal form, and the final optimal design cannot be identified until the subsystem boundary conditions approach a form associated with the final optimal design. In other words, the optimal design and the subsystem boundary conditions are interdependent, and they must be codetermined. Using the previous notation, we may denote the subsystem design variables as $\mathbf{x}^{\mathbf{i}}$, and the subsystem boundary conditions represent the global variables, **z**.

A direct iterative approach has been devised to solve this problem without the need for calculating sensitivity derivatives. In its simplest form, the algorithm works as shown in Figure 2. Note that the subsystem optimization in step 3 is typically terminated prior to convergence to the optimal solution. Often, there is no point in expending the extra effort toward finding an exact subsystem optimum prior to identifying subsystem boundary conditions that are close to their final form. Hence, the iterative process often proceeds using near optimal subsystem solutions.

It has been found that convergence toward an improved solution is greatly improved when the optimized subsystem design(s) in iteration *k*:

1. have good performance under the k^{th} set of subsystem boundary conditions; and
2. exhibit similar performance characteristics under the k^{th} and $(k+1)^{th}$ boundary conditions (i.e., the system and subsystem solutions in steps 3 and 4 (see Figure 2), respectively, do not have significantly different gradients or eigenmodes).

Thus, it is important during intermediate iterations to identify optimal or near optimal subsystem designs that have similar performance under small to moderate variations in the subsystem boundary conditions. Optimal subsystem designs that satisfy the above criteria are said to be robust against stochastic variations in the subsystem boundary conditions.

Convergence may also be improved by reducing the magnitude of the change in subsystem boundary conditions from one iteration to the next, or by using a weighted average of the boundary conditions at two consecutive steps.

In order to satisfy the conditions above, the subsystem boundary conditions at iteration *k* can be cast in the form:

$$\hat{u}_{subsystem(i)} = \sum_{s=1}^{k} \tilde{w}_s \hat{u}_s \qquad \text{on } \Gamma_{subsystem(i)} \tag{3}$$

where $\hat{u}_s(x,y,z,t)$ are the subsystem boundary conditions in terms of generalized primary variables on $\Gamma_{subsystem(i)}$ at iteration *s*; and $\tilde{w}_s(x,y,z,t)$ are weight functions whose spatial and temporal distributions are predetermined and whose magnitudes are varied stochastically within a selected range. Note that making the boundary conditions stochastic in this procedure gives rise to a very small number of stochastic variables, so that the computational cost penalty associated with extra evaluations is also very small.

It is possible for the interaction between the system and a subsystem to be specified or constrained along a portion of the boundary $\Gamma_{subsystem(i)}$, whenever this interaction is either known or desired to be of a particular form.

COMMENTS ON THE CURRENT APPROACH

The current strategy is independent of the optimization search algorithm used to optimize any subsystem. Moreover, it is possible to use entirely different optimization search algorithms to optimize different subsystems.

The system and subsystem models need not be of the same type. The only requirement is that the interactions between the system and subsystem can be identified, with sufficient accuracy to make the subsystem analyses meaningful, by performing an evaluation of the system model. The use of different model types for the system and subsystem can be of significant advantage in some cases.

Though the description here refers to subsystems in the spatial domain, it is also possible to apply this decomposition approach to the domains of time, design space, and scientific discipline. It is further possible for multiple types of decomposition to exist in the same decomposed system.

The procedure described here can be generalized to the case in which the system is hierarchically decomposed into N subsystems, wherein the ith subsystem is further decomposed into N_i subsystems, each of which is further decomposed into N_{ij} subsystems, and so on.

BENEFITS OF THE CURRENT APPROACH

When applying existing approaches to design problems in which each evaluation is very expensive, engineers can afford to perform only a small number of design iterations due to time and resource constraints. Hence, only a small number (2-5) of design parameters can be considered. The main advantage of the current approach is that thousands of design iterations can be performed in about the same amount of time required to perform 5-15 system level evaluations. Therefore, hundreds of design parameters can be considered simultaneously. With this approach, engineers can achieve designs with significantly higher performance and robustness, and in less time. Furthermore, engineers can thoroughly explore new design concepts that were previously ignored due to their computational complexity.

IMPLEMENTATION

COMPOSE is a strategy for design optimization that is generally independent of optimization search algorithm, but it does require a rather sophisticated process integration capability in order to perform effectively. For this reason, it has been implemented in such a way as to take advantage of the unique multi-agent architecture available within the design optimization software package HEEDS.

HEEDS (Hierarchical Evolutionary Engineering Design System) allows designers to automatically and concurrently explore hundreds of design parameters and their relationships in product and process design scenarios, and intelligently seeks optimal values for parameters that affect performance and cost. It can be used to improve any engineering system (structural, thermal, fluid, electrical, etc.), including multi-disciplinary problems. In structural design applications, for example, HEEDS can evolve designs that simultaneously satisfy objectives and targets for stiffness, durability, crashworthiness, noise and vibration, mass, cost, manufacturability and reliability.

HEEDS applies several optimization search processes simultaneously, allowing each process to take advantage of the best attributes and solutions found from other parallel searches. The multiple semi-independent search processes exchange information about the solution space with each other, helping them jointly to satisfy multiple constraints and objectives. This search method is called a *heterogeneous multi-agent approach*. It quickly identifies design attributes with good potential and uses them to focus, improve and accelerate the search for an optimum solution.

The HEEDS multi-agent approach also allows multiple adaptive hybrid methods to work semi-independently on a common problem. These methods facilitate decomposition of a design problem so that the search can be performed on various representations (e.g., different numbers of design variables at different levels of refinement) of the problem and using different performance measures (e.g., combinations of objectives and constraints) and local search methods. This optimization strategy efficiently searches for good designs at lower levels of refinement, developing “hunches” that can guide and be shared with other agents that are simultaneously searching with the more detailed representations required for final results. Use of multiple agents allows this multi-level approach to be performed on multiple computer processors to further speed the search, but it can also be implemented on a single processor. In general, design iterations that formerly required several months are now completed in days or hours.

AN EXAMPLE APPLICATION

The system (truck) and subsystem (lower compartment rails) shown in Figure 3 were selected for demonstrating the application of COMPOSE to crashworthiness design problems. In this problem, the vehicle impacts a rigid wall with an initial velocity of 35 mph. The shape of the lower compartment rails was designed using 140 shape design parameters, 70 in each rail. The design parameters were actually spline points which determined the cross-sectional shape of the rail at various stations along its length, as shown in Figures 3(c,d) and 4. An automatic mesher was used to generate a new mesh for each potential design. Each rail was designed separately. The system and subsystem level finite element analyses were performed using LS-DYNA, an explicit finite element code. Boundary conditions at the system/subsystem interface were extracted from the system level model and then imposed on the subsystem level model. The automated design optimization was executed on a personal computer during a period of about five days. The energy absorbed in the subsystem (rails) was increased by approximately 30% (see Figure 5(a)), while the overall energy absorbed by the system (truck) was increased by more than 5.5% (see Figure 5(b)). In Figure 5(a), the curve denoted Local EA represents the increase in energy absorbed in the rails as measured by subsystem model, while the curve denoted Global EA represents the increase in energy absorbed in the rails as measured by the system level model. These results differ slightly due to the multiple contact conditions that occur in the system level model, which change as the rail design is modified. Only six system level evaluations were performed, but complete coupling between the system and subsystem was maintained. This application clearly demonstrates the potential of COMPOSE to solve crashworthiness problems and other classes of design problems that were formerly considered intractable.

REFERENCES

[1] Averill, R.C., W. F. Punch, E. D. Goodman, S.-C. Lin, Y. C. Yip, Y. Ding, 1995, "Genetic Algorithm-Based Design of Energy Absorbing Laminated Composite Beams", *Proc. ASME Design Eng. Tech. Conf.,* Vol. 1, Boston, pp. 89-96.

[2] B. Malott, R. C. Averill, E. D. Goodman, Y. Ding, W. F. Punch, 1996, "Use of Genetic Algorithms for Optimal design of Laminated Composite Sandwich Panels with Bending-Twisting Coupling", *AIAA/ASME/ ASCE/AHS/ASC 37th Structures, Structural Dynamics and Materials Conference*, Salt Lake City, Utah.

[3] D. Eby, R. C. Averill, W. Punch, O. Mathews, E. Goodman, 1997, "An Island Injection GA for Flywheel Design Optimization", *Proc. EUFIT '97*, Aachen, Germany, pp. 687-691.

[4] D. Eby, R. Averill, W. Punch, E. Goodman, "Evaluation of Injection Island GA Performance on Flywheel Design Optimization," *Adaptive Computing in Design and Manufacture,* I. C. Parmee, ed., Springer, Berlin, 1998, pp. 121-136.

[5] Red Cedar Technology, proprietary internal report, 2001.

[6] R.C. Averill, D. Eby and E. Goodman, "How Well Can It Take a Hit? – An advanced automated optimization technique can help designers develop crashworthy structures." ASME Mechanical Engineering Design, pp. 26-28, March, 2001.

[7] C.A. Soto and A.R. Diaz, 1993, "Optimum layout and shape of plate structures using homogenization," in *Topology Design of Structures*, M.P. Bendsoe and C.A. Mota Soares, eds., pp. 407-420.

[8] K. Suzuki and N. Kikuchi, 1990, "Shape and topology optimization by a homogenization method," in *Sensitivity Analysis and Optimization with Numerical Methods*, AMD-Vol. 115, ASME, pp. 15-30.

[9] K. Suzuki and N. Kikuchi, 1991, "A homogenization method for shape and topology optimization," *Comp. Meth. Appl. Mech. Eng.*, **93**, 291-318.

[10] E. Sandgren, E. Jensen, and J.W. Welton, 1990, "Topological design of structural components using genetic optimization methods," in *Sensitivity Analysis and Optimization with Numerical Methods*, S. Saigal and S. Mukherjee, eds., AMD-Vol. 115, ASME, pp. 31-43.

[11] C.D. Chapman, K. Saitou, and M.J. Jakiela, 1993, "Genetic algorithms as an approach to configuration and topology design," in Proc. *1993 ASME Design Automation Conference*, Albuquerque, New Mex., Sept.

[12] S. Nagendra, R.T. Haftka, and Z. Gurdal, 1992, "Stacking sequence optimization of simply supported laminates with stability and strain constraints," *AIAA Journal*, **30**, 2132-2137.

[13] R. LeRiche and R.T. Haftka, 1993, "Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm," *AIAA Journal*, **31**, 951-956.

[14] S. Nagendra, R.T. Haftka, and Z. Gurdal, 1993, "Design of blade stiffened composite panels by a genetic algorithm approach," in *Proceedings of the 34th AIAA/ASME/AHS SDM Conference*, La Jolla, CA, April 19-22, pp. 2418-2436.

[15] M. Leung and G.E. Nevill, Jr., 1994, "Genetic algorithms for preliminary 2-D structural design," in *Proceedings of the 35th AIAA/ASME/AHS SDM Conference*, Hilton Head, SC, April 18-20.

[16] S.-C. Lin, W.F. Punch, and E.D. Goodman, 1994, "Coarse-grain parallel genetic algorithms: categorization and analysis," IEEE Symposium on Parallel and Distributed Processing, pp.27-36.

[17] Process Integration, Design Optimization Emerging as New Solution Category, by Daratech Research Staff, 2002.

[18] D. Eby, R.C. Averill, E.D. Goodman and W. Punch, "The Optimization of Flywheels Using an Injection Island Genetic Algorithm," *Evolutionary Design by Computers*, P. Bentley, editor, Morgan Kaufmann, San Francisco, pp. 167-190, 1999.

[19] J. Sobieszczanski-Sobieski, B.B. James and A.R. Dovi, "Structural Optimization by Multilevel Decomposition," *AIAA Journal*, Vol. 23, No. 11, pp. 1775-1782, 1985.

[20] J. Sobieszczanski-Sobieski, B.B. James and M.F. Riley, "Structural Sizing by Generalized, Multilevel Optimization," *AIAA Journal*, Vol. 25, No. 1, pp. 139-145, 1987.

[21] R.T. Haftka, "An Improved Computational Approach for Multilevel Optimum Design," *J. Structural Mechanics,* Vol. 12, pp. 245-261, 1984.

[22] M. Papadrakakis and Y. Tsompanakis, "Domain Decomposition Methods for Parallel Solution of Shape Sensitivity Analysis Problems," *Int. J. Num. Meth. Eng.,* Vol. 44, pp. 281-303, 1999.

[23] J. Sobieszczanski-Sobieski, T.D. Altus, M. Phillips and R. Sandusky, "Bilevel Integrated System Synthesis for Concurrent and Distributed Processing," *AIAA Journal*, Vol. 41, pp. 1996-2003, 2003.

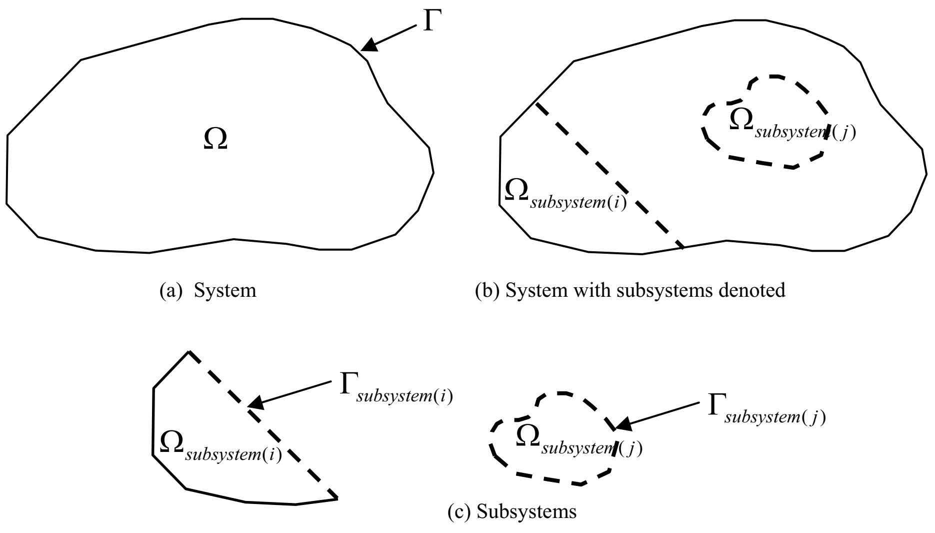


Figure 1. Schematic of system and subsystem domains and their boundaries.

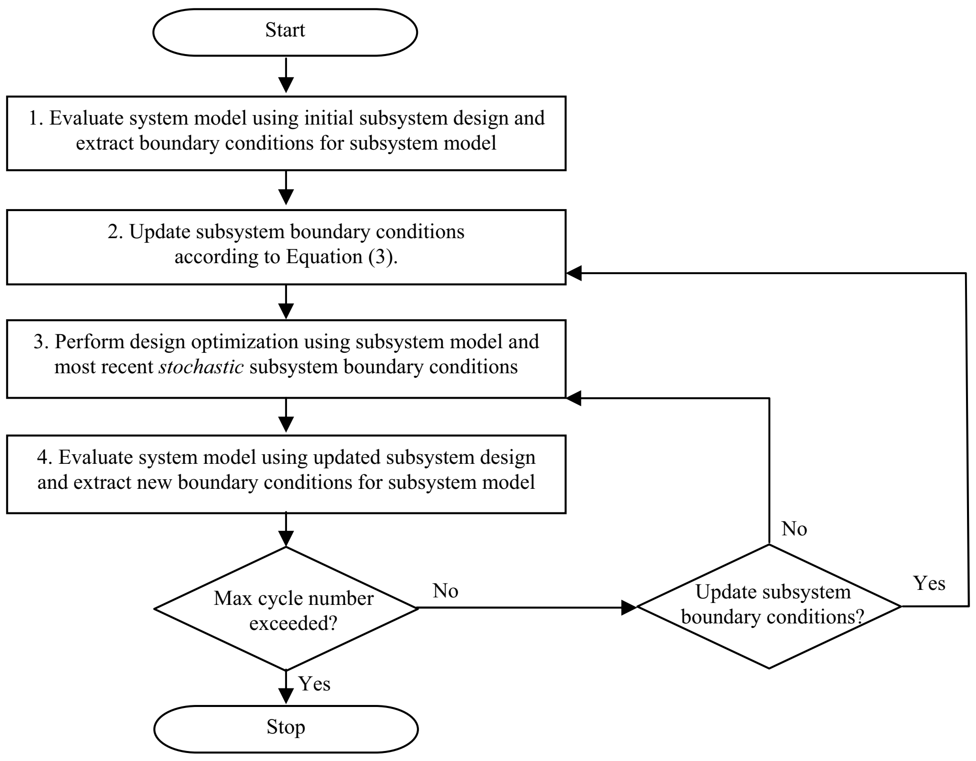
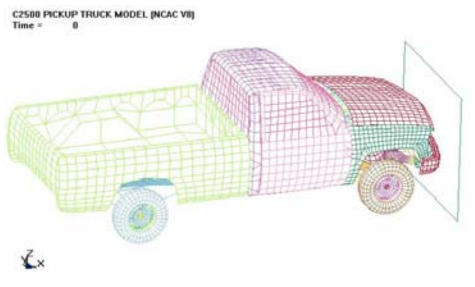
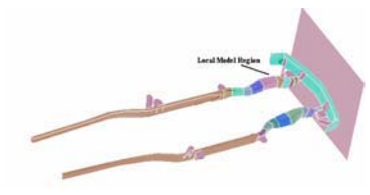

Start
1. Evaluate system model using initial subsystem design and extract boundary conditions for subsystem model
2. Update subsystem boundary conditions according to Equation (3).
3. Perform design optimization using subsystem model and most recent *stochastic* subsystem boundary conditions
4. Evaluate system model using updated subsystem design and extract new boundary conditions for subsystem model
Max cycle number exceeded?
No
Update subsystem boundary conditions?
No
Yes
Yes
Stop

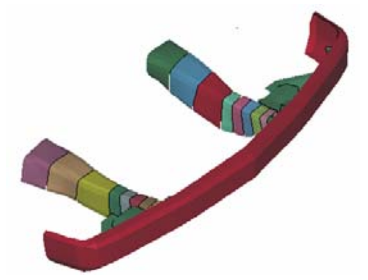
Figure 2. Flow chart for the COMPOSE iteration strategy to solve multi-level optimization problems.



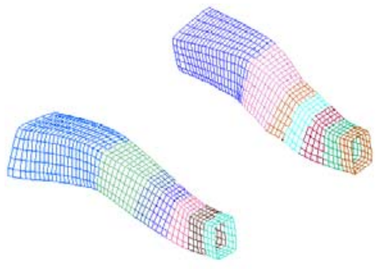
(a) System Level Model (pickup truck)



(b) Subsystem Level Model (lower rails)



(c) Reduced Subsystem Level Model



(d) Finite element model of subsystem rails

Figure 3. Crash model used to demonstrate COMPOSE.

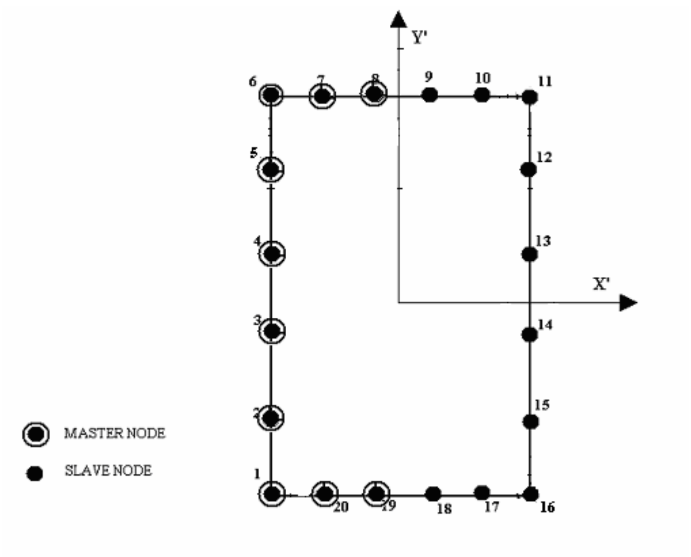
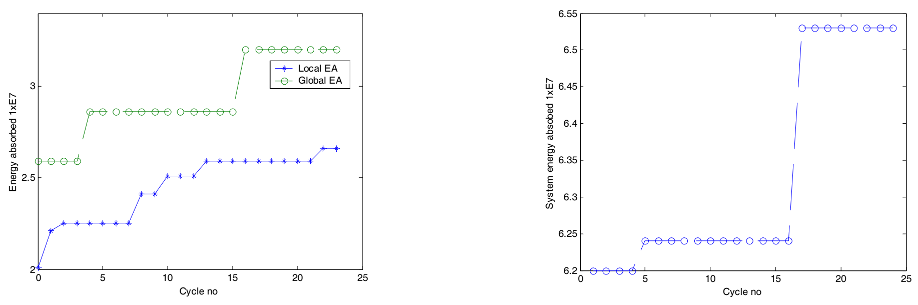


Figure 4. Cross-sectional shape representation. Each node represents a spline point that can move normal to the original rectangular shape in the X'Y' plane.



(a) (b)

Figure 5. Energy absorbed in subsystem (a) and system (b) as a function of design cycle.