Several parameters have been used to correlate fatigue crack growth rates under mixed-mode conditions such as equivalent stress intensity factors, equivalent strain intensity factors, strain energy density, and the *J*-integral. Fatigue cracks can also deflect in new directions and branch into multiple cracks under mixed-mode loading conditions. References 2 and 3 discuss these complex fatigue crack growth issues in detail.

References

1. Collins, J. 1991. *Failure of Materials in Mechanical Design*, 2nd ed., Wiley Interscience.
2. Suresh, S. 1991. *Fatigue of Materials*. Cambridge University Press, Cambridge, U.K.
3. Stephens, R.I., Fatemi, A., Stephens, R.R., and Fuchs, H.O. 2001. *Metal Fatigue in Engineering*, 2nd ed., Wiley Interscience, New York.
4. Norton, R.L. 1996. *Machine Design*. Prentice Hall, Englewood Cliffs, NJ.
5. Stouffer, D.C. and Dame, L.T. 1995. *Inelastic Deformation of Metals: Models, Mechanical Properties, and Metallurgy.* Wiley Interscience, New York.
6. Dowling, N.E. 1993. *Mechanical Behavior of Materials*. Prentice-Hall, Englewood Cliffs, NJ.
7. Progress in Measuring Fracture Toughness and Using Fracture Mechanics, *Materials Research Standards*, ASTM (March 1964): 103–19.
8. Tada, H., Paris, P.C., and Irwin, G.E. 1985 *The Stress Analysis of Cracks Handbook*, 2nd ed. Paris Productions, Inc., St. Louis, MO.
9. Wöhler, A. Versuche über die Festigkeit der Eisenbahnwagenachsen, *Zeitschrift für Bauwesen*, Vol. 10, 1860; English summary (1867), *Engineering*, Vol. 4, 160-1.
10. Neuber, H., Theory of stress concentration for shear-strained prismatical bodies with arbitrary nonlinear stress-strain law. *J. of Appl. Mech.*, ASME Transactions, Vol. 8, pp. 544–50, 1961.
11. Fatemi, A. and Socie, D.F., A critical plane approach to multiaxial fatigue damage including out-of-phase loading, *Fatigue Fract. Eng. Mater. Struct.*, Vol. 11, No. 3, 1988, p. 149.
12. Paris, P.C., Gomez, M.P., and Anderson, W.P., A rational analytic theory of fatigue, *The Trend in Engineering*, Vol. 13, pp. 9–14, 1961.
13. Paris, P.C. and Erdogan, F., A critical analysis of crack propagation laws, *J. of Basic Eng.*, Vol. 85, pp. 528–34, 1963.

11.5 Design Optimization

Nam Ho Kim

Introduction

The design of a structural system has two categories: designing a new structure and improving the existing structure to perform better. The design engineer's experience and creative ideas are required in the development of a new structure, since it is difficult to quantify a new design using mathematical measures. Recently, limited inroads have been made in the creative work of the structural design using mathematical tools.[1] However, the latter evolutionary process is encountered much more frequently in engineering designs. For example, how many times does an automotive company design a new car using a completely different concept? The majority of a design engineer's work concentrates on improving the existing vehicle so that the new car can be more comfortable, more durable, and safer. In this section, we will focus on a design's evolutionary process by using mathematical models and computational tools.

Structural design is a procedure for improving or enhancing the performance of a structure by changing its parameters. A *performance measure, which* can be quite general in engineering fields, can include the following: the weight, stiffness, and compliance of a structure; the fatigue life of a mechanical component; the noise in the passenger compartment; the vibration level; the safety of a vehicle in a crash, and so forth. However, this does not address such aesthetic measures as whether a car or a structural design is attractive to customers. All performance measures are presumed to be measurable quantities. System parameters are variables that a design engineer can change during the design process. For example, the thickness of a vehicle body panel can be changed to improve vehicle performance. The cross section of a beam can be changed in designing bridge structures. System parameters that can be changed during the design process are called *design variables*, even including the geometry of the structure.

Great strides have been made during the past decade in computer-aided design (CAD) and computer-aided engineering (CAE) tools for mechanical system development. Discipline-oriented simulation capabilities in structures, mechanical system dynamics, aerodynamics, control systems, and numerous related fields are now being used to support a broad range of mechanical system design applications. Integration of these tools to create a robust simulation-based design capability, however, remains a challenge. Based on their extensive survey of the automotive industry in the mid-1980s, Clark and Fujimoto[2] concluded that simulation tools in support of vehicle development were on the horizon but not yet ready for pervasive application. The explosion in computer, software, and modeling and simulation technology that has occurred since the mid-1980s suggests that high-fidelity tools for simulation-based design are now at hand. Properly integrated, they can resolve uncertainties and significantly affect mechanical system design.

Modern developments of structural design are closely related to concurrent engineering environments by which multidisciplinary simulation, design, and manufacturing are possible. Even though concurrent engineering is not the focus of this section, we want to emphasize structural design as a component of concurrent engineering. An important feature of the concurrent engineering is database management using the CAD tool. Structural modeling and most interfaces are achieved using the CAD tool. Thus, design parameterization and structural model updates have to be carried out within the CAD model. Through the design parameterization, CAD, CAE, and CAM procedures are interrelated to form a concurrent engineering environment.

The structural engineering design in the simulation-based process consists of structural modeling, design parameterization, structural analysis, design problem definition, design sensitivity analysis, and design optimization. Figure 11.5.1 is a flowchart of the structural design process in which computational analysis and mathematical programming play essential roles in the design. The success of the system-level, simulation-based design process strongly depends on a consistent design parameterization, an accurate structural and design sensitivity analysis, and an efficient mathematical programming algorithm.

A design engineer simplifies the physical engineering problem into a mathematical model that can represent the physical problem up to the desired level of accuracy. A mathematical model has parameters that are related to the system parameters of the physical problem. A design engineer identifies those design variables to be used during the design process. *Design parameterization*, which allows the design engineer to define the geometric properties for each design component of the structural system being designed, is one of the most important steps in the structural design process. The principal role of design parameterization is to define the geometric parameters that characterize the structural model and to collect a subset of the geometric parameters as design variables. Design parameterization forces engineering teams in design, analysis, and manufacturing to interact at an early design stage, and supports a unified design variable set to be used as the common ground to carry out all analysis, design, and manufacturing processes. Only proper design parameterization will yield a good optimum design, since the optimization algorithm will search within a design space that is defined for the design problem. The design space is defined by the type, number, and range of design variables. Depending on whether it is a concept or detailed design, selected design variables could be non-CAD based parameters. An example of such a design variable is a tire's stiffness characteristic in vehicle dynamics during an early vehicle design stage.

Structural analysis can be carried out using experiments in actual or reduced scale, which is a straightforward and still prevalent method for industrial applications. However, the expense and inefficiency involved in fabricating prototypes make this approach difficult to apply. The analytical method may resolve these difficulties, since it approximates the structural problem as a mathematical model and solves

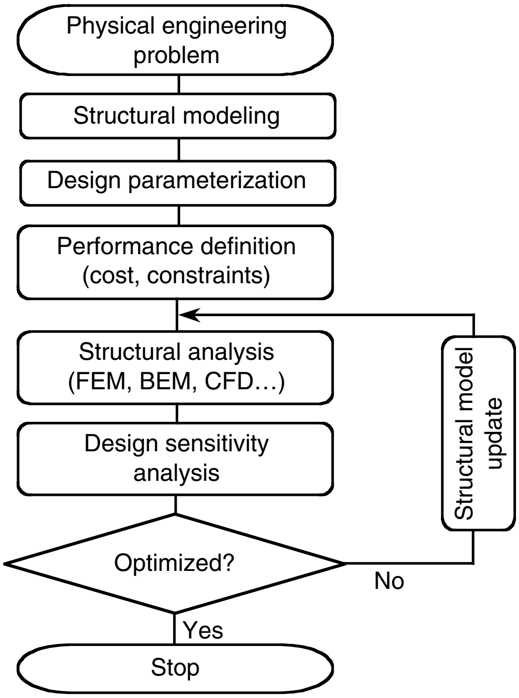

Physical engineering problem
Structural modeling
Design parameterization
Performance definition (cost, constraints)
Structural analysis (FEM, BEM, CFD…)
Design sensitivity analysis
Optimized?
No
Structural model update
Yes
Stop

FIGURE 11.5.1 Structural design process.

it in a simplified form. The mathematical model can be used to evaluate the performance measures of a structural problem. However, the analytical method has limitations even for simple structural problems.

With the emergence of various computational capabilities, most analytical approaches to mathematical problems have been converted to numerical approaches, which are able to solve very complicated, real engineering applications. Finite element analysis (FEA), boundary element analysis (BEA), and mesh-free analysis are a short list of mathematical tools used in structural analysis. The development of FEA is one of the most remarkable successes in structural analysis. The governing differential equation of the structural problem is converted to its integral form and then solved using FEA. Vast amounts of literature are published regarding FEA; for example, refer to reference 3 and the references therein. The complex structural domain is discretized by a set of nonoverlapping, simple-shaped finite elements, and an equilibrium condition is imposed on each element. By solving a linear system of matrix equations, we can compute the performance measures of a structure in the approximated domain. The accuracy of the approximated solution can be improved by reducing the size of finite elements and/or increasing the order of approximation within an element.

Selection of a design space and an analysis method must be carefully determined since the analysis, both in terms of accuracy and efficiency, must be able to handle all possible designs in the chosen design space. That is, the larger the design space, the more sophisticated the analysis capability must be. For example, if larger shape design changes are expected during design optimization, mesh distortion in FEA could be a serious problem and a finite element model that can handle large shape design changes must be used.

A *performance measure* in a simulation-based design is the result of structural analysis. Based on the evaluation of analysis results, such engineering concerns as high stress, clearance, natural frequency, or mass can be identified as performance measures for design improvement. Typical examples of performance measures are mass, volume, displacement, stress, compliance, buckling, natural frequency, noise, fatigue life, and crashworthiness. A definition of performance measures permits the design engineer to specify the structural performance from which the sensitivity information can be computed.

Cost and constraints can be defined by combining certain performance measures with appropriate constraint bounds for interactive design optimization. The *cost function*, sometimes called the *objective function*, is minimized (or maximized) during optimization. Selection of a proper cost function is an important decision in the design process. A valid cost function has to be influenced by the design variables of the problem; otherwise, it is not possible to reduce the cost by changing the design. In many situations, an obvious cost function can be identified. In other situations, the cost function is a combination of different structural performance measures. This is called a *multiobjective* cost function.

Constraint functions are the criteria that the system has to satisfy for each feasible design. Among all design ranges, those that satisfy the constraint functions are candidates for the optimum design. For example, a design engineer may want to design a bridge whose weight is minimized and whose maximum stress is less than the yield stress. In this case, the cost function, or weight, is the most important criterion to be minimized. However, as long as stress, or constraint, is less than the yield stress, the stress level is not important.

Design sensitivity analysis is used to compute the sensitivity of performance measures with respect to design variables. This is one of the most expensive and complicated procedures in the structural optimization process. Structural design sensitivity analysis is concerned with the relationship between design variables available to the engineer and the structural response determined by the laws of mechanics. Design sensitivity information provides a quantitative estimate of desirable design change, even if a systematic design optimization method is not used. Based on the design sensitivity results, a design engineer can decide on the direction and amount of design change needed to improve the performance measures. In addition, design sensitivity information can provide answers to "what if" questions by predicting performance measure perturbations when the perturbations of design variables are provided.

Substantial literature has emerged in the field of structural design sensitivity analysis.[4] Design sensitivity analysis of structural systems and machine components has emerged as a much-needed design tool, not only because of its role in optimization algorithms but also because design sensitivity information can be used in a computer-aided engineering environment for early product trade-off in a concurrent design process.

Recently, the advent of powerful graphics-based engineering workstations with increasing computational power has created an ideal environment for making interactive design optimization a viable alternative to more monolithic batch-based design optimization. This environment integrates design processes by letting the design engineer create a geometrical model, build a finite element model, parameterize the geometric model, perform FEA, visualize FEA results, characterize performance measures, and carry out design sensitivity analysis and optimization.

Design sensitivity information can be used during a postprocessing of the interactive design process. The principal objective of the postprocessing design stage is to utilize the design sensitivity information to improve the design. Figure 11.5.2 shows the four-step interactive design process: (1) to visually display design sensitivity information, (2) to carry out what-if studies, (3) to make trade-off determinations, and (4) to execute interactive design optimization. The first three design steps, which are interactive modes of the design process, help the design engineer improve the design by providing structural behavior information at the current design stage. The last design step, which could be either interactive or a batch mode of the design process, launches a mathematical programming algorithm to perform design optimization. Depending on the design problem, the design engineer could use some or all of the four design steps to improve the design at each iterative step. As a result, new designs could be obtained from what-if, trade-off, or interactive optimization design steps.

For the purposes of design optimization, a mathematical programming technique is often used to find an optimum design that can best improve the cost function within a feasible region. Mathematical programming generates a set of design variables that require performance values from structural analysis and sensitivity information from design sensitivity analysis to find an optimum design. Thus, the structural model has to be updated for a different set of design variables supplied by mathematical programming. If the cost function reaches a minimum with all constraint requirements satisfied, then an optimum design is obtained.

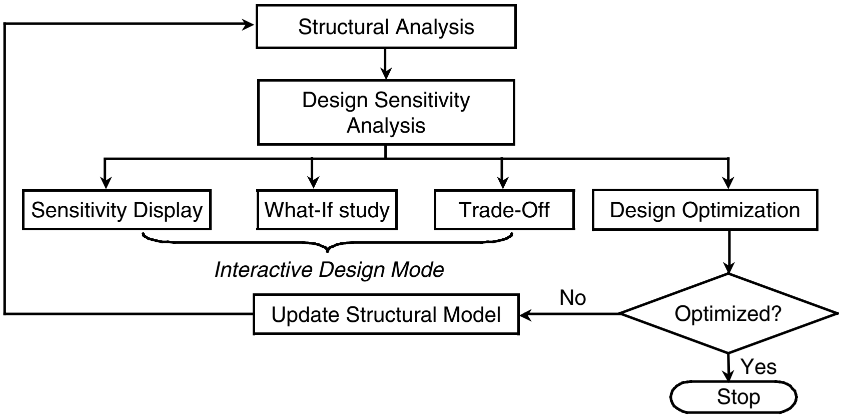


FIGURE 11.5.2 Postprocessing design stage.

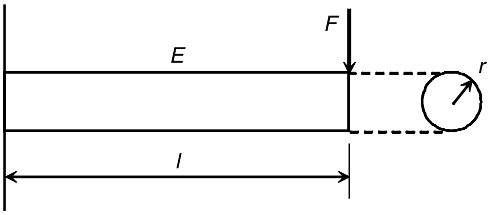


FIGURE 11.5.3 Parameters defining circular cross-sectional cantilever beam.

Structural Design Parameterization

In structural modeling, the physical problem is represented by mathematical expressions, which contain parameters for defining that problem. For example, the cantilever beam in Figure 11.5.3 has parameters that include the length l, the radius of cross section r, and Young's modulus E. These parameters, which define the system, are called *design variables*. If design variables are determined, then the structural problem can be analyzed. Obviously, different design variable values usually yield different analysis results. The aim of the structural design process is to find the values of design variables that satisfy all requirements.

All design variables must satisfy the physical requirements of the problem. For example, length l of the cantilever beam in Figure 11.5.3 cannot have a negative value. Physical requirements define the design variable bounds. Valid design variables may have to take into account various manufacturing requirements. For example, the radius of a cantilever beam satisfies its physical requirement if r is a positive number. However, in real applications, the circular cross-sectional beam may not be manufactured if its radius is bigger than r^0. Thus, the range of feasible design can be stated as $0 < r \le r^{01}$. In addition, the design engineer may want to impose certain design constraints on the problem. For example, the maximum stress of the beam may not exceed σ^0 and the maximum tip displacement of the beam must not be greater than z^0. A set of design variables that satisfy the constraints is called a *feasible design*, whereas a set that does not satisfy constraints is called an *infeasible design*. It is difficult to determine whether a current design is feasible unless the structural problem is analyzed. For complicated structural problems, it may not be simple to choose the appropriate design constraints so that the feasible region is not empty.

There are two types of design variables: continuous and discrete. Many design optimization algorithms consider design variables to be continuous. In this section, we presume that all design variables are

[1] In general, the bounds of design variables are denoted as $r^L \le r \le r^U$ where r^L is called the lower bound and r^U is called the upper bound, respectively.

continuous within their lower- and upper-bound limits. However, discrete design problems often appear in real engineering problems. For example, due to manufacturing limitations, the structural components of many engineering systems are only available in fixed shapes and sizes. Discrete design variables can be thought of as continuous design variables with constraints. As a result, it is more expensive to obtain an optimum design for a problem with discrete design variables. It is possible, however, to solve the problem assuming continuous design variables. After obtaining an optimum solution for the design problem, the nearest discrete values of the optimum design variables can be tested for feasibility. If the nearest discrete design variables are not feasible, then several iterations can be carried out to find the nearest feasible design.

It is convenient to classify design variables according to their characteristics. In the design of structural systems made of truss, beam, membrane, shell, and elastic solid members, there are five kinds of design variables: material property design variables such as Young's modulus; sizing design variables such as thickness and cross-sectional area; shape design variables such as length and geometric shape; configuration design variables such as orientation and location of structural components; and topological design variables.

The Material Property Design Variable

In structural modeling, the material property is used as a parameter of the structural problem. Young's modulus and Poisson's ratio, for example, are required in the linear elastic problem. If these material properties are subject to change, then they are called *material property design variables*. These kinds of design variables do not appear in regular design problems, since in most cases material properties are presumed to be constant. Analysis using such a constant material property is called the deterministic approach. Another approach uses probability and assumes that material properties are not constant but distributed within certain ranges. This is called the probabilistic approach and is more practical, since a number of experiments will usually yield a number of different test results. In this case, material properties are no longer considered to be constant and can therefore be used as design variables.

The Sizing Design Variable

The *sizing design variable* is related to the geometric parameter of the structure, and it is often called a parametric design variable. For example, most automotive and airplane parts are made from plate/shell components. It is natural that a design engineer wants to change the thickness (or gauge) of the plate/shell structure in order to reduce the weight of the vehicle. For a structural model, plate thickness is considered a parameter. However, the global geometry of the structure does not change. Plate thickness can be considered a sizing design variable. The sizing design variable is similar to the material property design variable in the sense that both variables change the parameters of the structural problem.

Another important type of sizing design variable is the cross-sectional geometry of the beam and truss. Figure 11.5.4 provides some examples of the shapes and parameters that define these cross sections. In the structural analysis of truss, for example, the cross-sectional area is required as a parameter of the problem. If a rectangular cross section is used, then the area would be defined as $A = b \times h$. Thus, without any loss of generality, b and h can be considered design variables of the design problem.

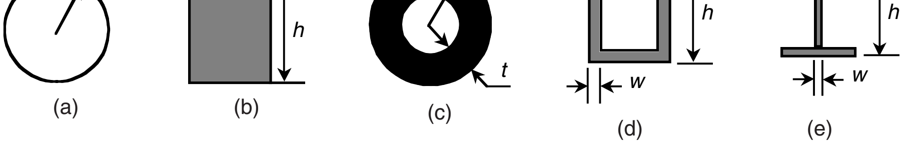


FIGURE 11.5.4 Sizing design variables for cross-sectional areas of truss and beam. (a) solid circular; (b) rectangular; (c) circular tube; (d) rectangular tube; (e) I-section.

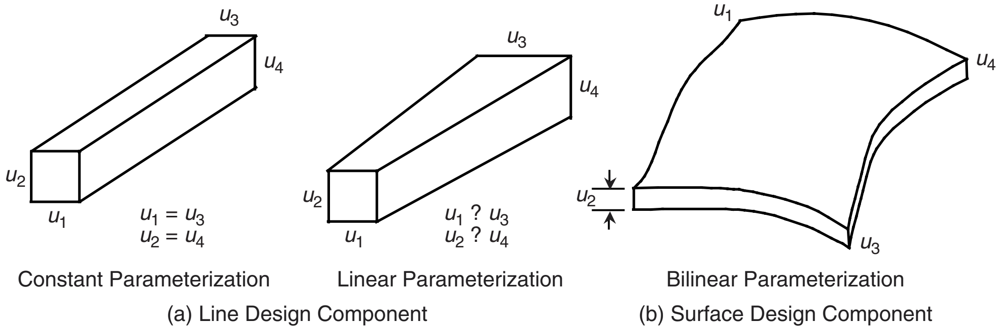


FIGURE 11.5.5 Line and surface design parameterization.

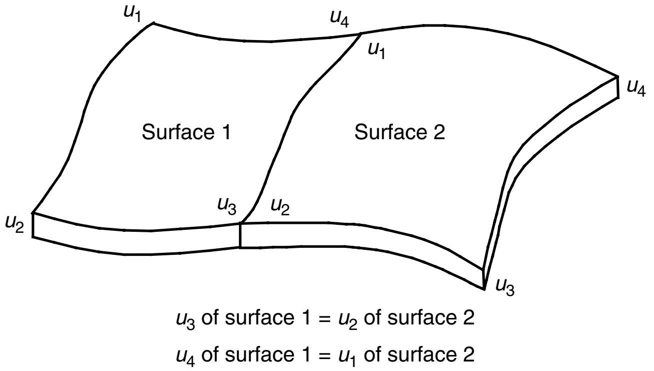

u_1
u_4
u_1
u_4
Surface 1
Surface 2
u_3
u_2
u_2
u_3
u_3 of surface 1 = u_2 of surface 2
u_4 of surface 1 = u_1 of surface 2

FIGURE 11.5.6 Design variable linking.

In this section, several possible design parameterizations, such as constant and linear designs as described in Figure 11.5.5, are introduced for the line and surface design components. These are not at all the only possible design parameters, and other more complicated design parameterizations can be used. However, the method presented in this section can be extended to other complicated design parameterizations. One important thing to consider when more complicated design parameterizations are used is that the finite element model must be sophisticated enough to support the design parameterization method used. Geometric parameters can be defined at the end grid points of a line, or at the corner points of a surface. A bilinear thickness distribution can be used to characterize a surface design component, as shown in Figure 11.5.5(b). Note that each dimension that defines the cross-sectional shape, such as width and height in Figure 11.5.5(a), can be treated as a design variable, and be allowed to vary to the same degree as the corresponding variable at the other end (constant parameterization), or to a different degree (linear parameterization). Moreover, in order to maintain design continuity for a symmetric design, or to reduce the number of design variables, design variables can change either independently of, or proportionally to, certain variables across design components through design variable linking, as shown in Figure 11.5.6.

Line Design Components

A three-dimensional line component can be used for truss or beam design components. A truss design component can handle tensile and compressive load and may be composed of several truss finite elements. A beam design component can handle tensile, compressive, and bending load. A linearly tapered cross-sectional shape can be considered within the design component. There are three cross-section types for this design component: symmetric, un-symmetric, and general. Figure 11.5.7 illustrates the geometry of a line design component.

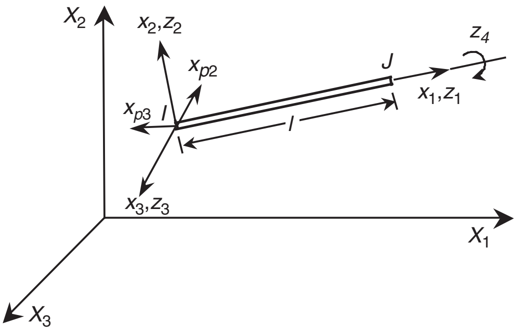
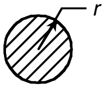
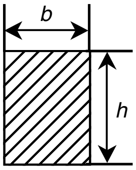
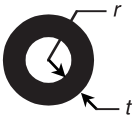
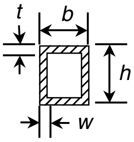
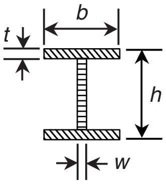


FIGURE 11.5.7 Line design component.

Material properties, such as mass density and Young's modulus, and geometric parameters that define the cross-sectional shape can be taken as design variables along the axial axis x_1. All material property design variables are assumed to be constant along the design component's axial axis.

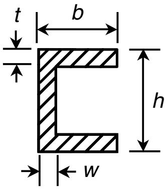
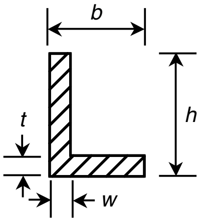
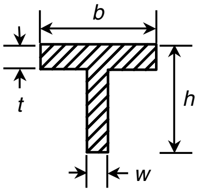
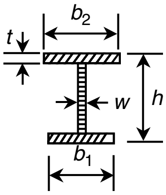
The geometric design variables that can linearly vary along the axis of the design component are the dimensions of each cross-section, that is, r for a solid and hollow circular cross section, and b and h for other cross sections, as shown in Table 11.5.1 and Table 11.5.2.

TABLE 11.5.1 Symmetric Cross Sections

Cross-Sectional Shape		Design Variable[a]					
		1	2	3	4	5	6
Solid circular		r	E	ρ	—	—	—
Solid rectangular		h	b	E	ρ	—	—
Hollow circular		r	t	E	ρ	—	—
Hollow rectangular		h	b	t	w	E	ρ
I-section		h	b	t	w	E	ρ

[a] E is Young's modulus and ρ is the mass density.

TABLE 11.5.2 Unsymmetric Cross Sections

		Design Variable						
Cross-Sectional Shape		1	2	3	4	5	6	7
Channel		*h*	*b*	*t*	*w*	*E*	ρ	—
Angle		*h*	*b*	*t*	*w*	*E*	ρ	—
T-section		*h*	*b*	*t*	*w*	*E*	ρ	—
Unsymmetric I-section		*h*	b_1	b_2	*t*	*w*	*E*	ρ

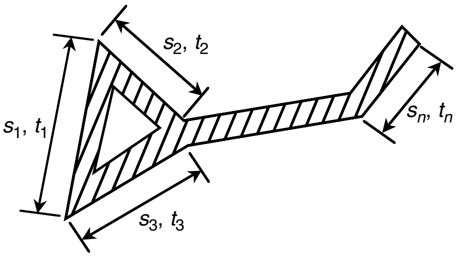
Proportionality can be used between two geometric design variables within a design component through design variable linking. Such proportional design variables are used to meet local buckling requirements for a structural design specification, to maintain geometric proportionality, or to reduce the number of design variables. The pairs of geometric design variables that may be considered for proportionality purposes are (t,b) and (w,h) for a hollow rectangle, an I-section, (b,h) for a solid rectangle, and (t,r) for a hollow circle.

The symmetric cross section has two perpendicular axes of symmetry, as shown in Table 11.5.1. These two axes are the principal axes of the cross section, and are usually taken as two component coordinates. The component's axial axis is assumed to be the same as the centroidal axis. Solid and hollow circles, solid and hollow rectangles, and the I-section are all placed in this category.

The unsymmetric cross section has either only one axis of symmetry, or no symmetry with some shape regularity, as shown in Table 11.5.2. The channel, angle, T-section, and unsymmetric I-section all fit into this category. The component coordinates x_2 and x_3 may be different from the principal axes x_{p2} and x_{p3}, respectively. The centroidal axis is assumed to coincide with the axial axis x_1 of the truss design component.

The general cross section has no regular shape and may be composed of several thin-walled segments, as shown in Table 11.5.3.

TABLE 11.5.3 General Cross Sections

	Design Variable					
General Cross-Sectional Shape	1	2	3	…	$2n+1$	$2n+2$
	s_1	t_1	s_1	…	E	ρ

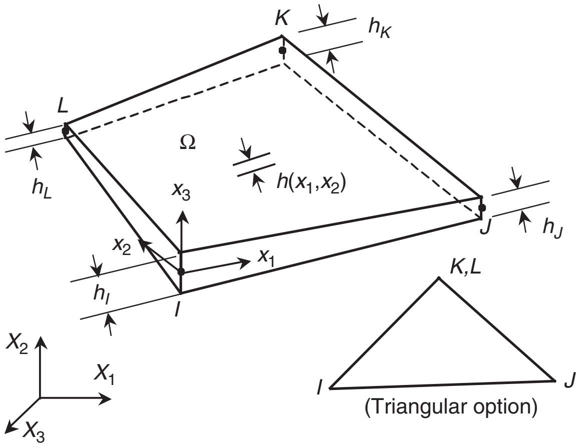


FIGURE 11.5.8 Surface design component.

Surface Design Components

A membrane component can handle both an in-plane tensile and a compressive load. A shell component can handle an in-plane tensile, compressive, and bending load. The design component may be composed of several membrane/shell finite elements. Figure 11.5.8 illustrates the geometry of a membrane/shell design component.

Surface component design variables include thickness, mass density, and Young's modulus. Surface thickness is parameterized using a bilinear shape function. Four geometric design variables are defined for each surface design component: thickness h_I, h_J, h_K, and h_L at grid points I, J, K, and L, respectively, as shown in Figure 11.5.8 and Table 11.5.4.

The four-node quadrilateral surface component can be reduced to a triangular surface component by defining duplicate node numbers for the third and fourth (K and L) node locations. If node L is not defined, then it defaults to node K. The design component thickness is assumed to vary bi-linearly inside the design component.

TABLE 11.5.4 Design Variables of Surface Design Component

Design Variables					
1	2	3	4	5	6
E	ρ	h_I	h_J	h_K	h_L

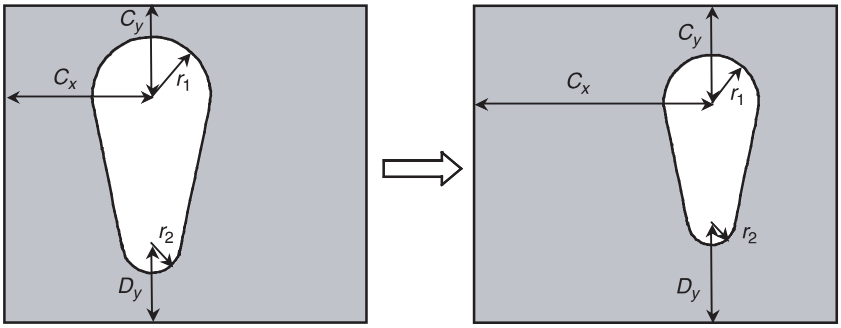


FIGURE 11.5.9 Shape design variables.

The Shape Design Variable

While material property and the sizing design variables are related to the parameters of the structural problem, the shape design variable is related to the structure's geometry. The shape of the structure does not explicitly appear as a parameter in the structural formulation. Although the design variables in Figure 11.5.4 determine the cross-sectional shape, they are not shape design variables, since these cross-sectional shapes are considered parameters in the structural problem. However, the length of the truss or beam should be treated as a shape design variable. Usually, the shape design variable defines the domain of integration in structural analysis. Thus, it is not possible to extract shape design variables from a structural model and to use them as sizing design variables.

Consider a rectangular block with a slot, as presented in Figure 11.5.9. The location and size of the slot is determined by the geometric values of C_x, C_y, D_y, r_1, and r_2, which are shape design variables. Different values of shape design variables yield different structural shapes. However, these shape design variables do not explicitly appear in the structural problem. If the finite element method is used to perform structural analysis, then integration is carried out over the structural domain (the gray area), which is the shape design variable. Since shape design variables do not explicitly appear in the structural problem, the shape design problem is more difficult to solve than the sizing design problem.

Shape design parameterization, which describes the boundary shape of a structure as a function of the design variables, is an essential step in the shape design process. Inappropriate parameterization can lead to unacceptable shapes.[5,6] To parameterize the structural boundaries and to achieve optimum shape design, boundary shape can be described in three ways: (1) by using boundary nodal coordinates, (2) by using polynomials, or (3) by using spline blending functions. However, it is important to point out that there are many methods of parameterization and that the methods presented in this section are only a few of them, including complicated parameterization methods developed in commercial CAD tools. One important aspect of shape design parameterization is the connection of the design parameterization to the computation of the design velocity field.

In the first method, boundary nodal coordinates of the finite element model are used as shape design variables. Although the method is simple and easy to use, it has the following drawbacks: (1) the number of design variables tends to become very large, which may lead to high computational costs and optimization problems that are difficult to solve; (2) the first derivative of the design boundary is not continuous across boundary nodes, which may lead to an unacceptable or impractical design; and (3) computational accuracy is not ensured, since it is difficult to maintain an adequate finite element mesh during the optimization process. One can use coordinates of selected master nodes as shape design variables and employ an isoparametric mapping to generate a finite element mesh.

Several methods have been developed to parameterize the design boundary with polynomials. Bhavikatti and Ramakrishnan[7] used a 5th-degree polynomial, with the coefficients taken as design variables, to parameterize the boundary shape of a rotating disk. Prasad and Emerson[8] used a similar approach to optimize an engine connecting rod. In a more general approach, such as that used by Kristensen and Madsen[9] and Pedersen and Laursen,[10] the boundary is parameterized using a linear combination of shape functions, with the coefficients as design variables. The total number of shape design variables can be reduced using polynomials for shape representation. However, using high-order polynomials to represent the boundary shape may result in oscillating boundaries.

Splines eliminate the problem of oscillating boundaries since they are composed of low-order polynomial segments that are combined to maximize the smoothness of the boundaries. Yang and Choi,[11] Luchi et al.,[12] and Weck and Steinke[13] used a cubic spline to define the boundary geometry. The spline representation was shown to yield better sensitivity accuracy than a piecewise linear representation of the boundary.[11] Braibant et al.[14] used Bezier and B-spline blending functions to describe the design boundary. Blending functions provide great flexibility for geometrical description. Using B-splines, Braibant and Fleury[15] optimized a beam in bending, a fillet, and a hole in a plate. Finally, Yao and Choi[16–18] used a Bezier surface to optimize an engine bearing cap and an arch dam.

The shape design parameterization method presented in this section deals with geometric features. A geometric feature is a subset of the geometric boundaries of a structural component. For example, a fillet or a circular hole is a geometric feature that contains characteristics associated with it and is likely to be chosen as a design variable. A geometric feature whose design variables are defined is known as a parameterized geometric feature, and is treated as a single entity in the shape design process. For example, a circular hole with its radius and center location defined as design variables is a parameterized geometric feature. Such a parameterized circular hole can be moved around in the structure with a varied size due to design changes. However, the shape of the circular hole itself remains constant.

Two steps are involved in the design parameterization process: geometric modeling and defining the design variables. A geometric model is first generated in the modeling process, with all its dimensions defined. Geometric features that can be varied in the design process need to be identified by both design and manufacturing engineers at the beginning of the design process. The design engineer then parameterizes the geometric model, using the geometric feature that is consistent with both engineering requirements and manufacturing limitations. The design parameterization developed in this section is based on the assumption that the geometric model has already been created and its dimensions defined.

In general, structural shape design problems can be classified into four types, in terms of the characteristics of the design boundary. In the first type, the shape of an arbitrary open or closed boundary is determined, such as a fillet[4] or a dam surface.[16] In the second type, the dimensions of predefined shapes are determined, such as the radius of a circular hole, the major and minor axes of an elliptic hole, the dimensions of a slot, the length of a rectangular membrane, or the radius of a rounded corner. In the third type, the design boundary location is determined, such as the center location of a circular hole, an elliptic hole, an arbitrarily shaped hole, or a slot. In the final type, a rotation angle of the design boundary, either arbitrary or predefined, is treated as the design variable. In this section, shape design parameterization of the first three types of design problems is considered.

In general, geometric entities can be represented using parametric cubic (PC) lines, patches (surfaces), and hyperpatches (solids). A parametric cubic line is represented by three functions:

$$\left.\begin{aligned} x &= x(u) \\ y &= y(u) \\ z &= z(u) \end{aligned}\right\} \tag{11.5.1}$$

where u is the parametric direction of the line with domain.[1] Each of these functions is, at most, a cubic polynomial of the form

$$z(u) = s_3 u^3 + s_2 u^2 + s_1 u + s_0 \tag{11.5.2}$$

The first and second derivatives of Equation 11.5.2 can be written as

$$z_{,u}(u) = 3s_3u^2 + 2s_2u + s_1 \tag{11.5.3}$$

$$z_{,uu}(u) = 6s_3u + 2s_2 \tag{11.5.4}$$

From Equation 11.5.3 and Equation 11.5.4, note that the slope of the cubic line can change its sign only twice, and that the curve can have only one inflection point. Consequently, PC entities such as PC lines and PC patches minimize the possibility of yielding oscillating boundaries during the design process.[6] However, geometric entities with predefined or sophisticated shapes, such as a circular hole, cannot be represented by a single cubic curve. To minimize modeling errors, such a boundary can be broken into small pieces. These pieces are then "glued" together in the design process as one geometric feature by appropriately linking design variables. For shape design, planar parametric cubic lines and spatial parametric bicubic patches are utilized to represent the design boundaries of two-dimensional and three-dimensional structural components, respectively.

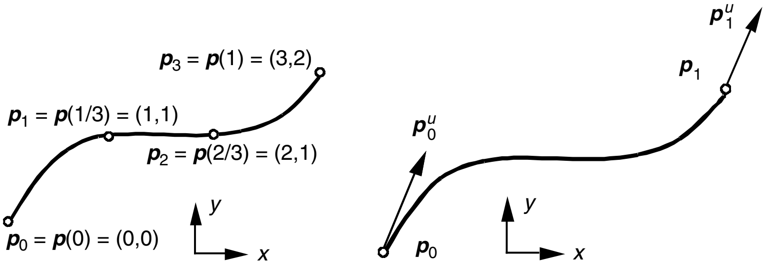
There are a number of methods for creating geometric entities, for example, defining four control points to create a Bezier curve, or constructing four edge curves to create a surface. Although geometric entities have different characteristics depending on the way they are created, they are nevertheless always represented by polynomials with the same order regardless of the way they are created. Consequently, a single geometric entity can be created using a variety of methods. For example, the planar curve shown in Figure 11.5.10(a) is created by defining four distinct points in the plane, and is thus called a four-point curve.

The mathematical expression of the four-point curve is given as

$$\left.\begin{aligned} x(u) &= 3u \\ y(u) &= 9u^3 - 13.5u^2 + 6.5u \end{aligned}\right\} \tag{11.5.5}$$

The same curve shown in Figure 11.5.10(b) can also be created by giving the position and slope at the endpoints, referred to as geometric coefficients:

$$\left.\begin{aligned} \mathrm{p}_0 &= [0,0] \\ \mathrm{p}_1 &= [3,2] \\ \mathrm{p}_0^u &= [3,6.5] \\ \mathrm{p}_1^u &= [3,6.5] \end{aligned}\right\} \tag{11.5.6}$$


$\boldsymbol{p}_3 = \boldsymbol{p}(1) = (3,2)$
$\boldsymbol{p}_1 = \boldsymbol{p}(1/3) = (1,1)$
$\boldsymbol{p}_2 = \boldsymbol{p}(2/3) = (2,1)$
$\boldsymbol{p}_0 = \boldsymbol{p}(0) = (0,0)$
y
x
$\boldsymbol{p}_1^u$
$\boldsymbol{p}_1$
$\boldsymbol{p}_0^u$
$\boldsymbol{p}_0$
y
x

(a) (b)

FIGURE 11.5.10 Planar parametric cubic curves (a) curve created by four points; (b) curve created by geometric coefficients.

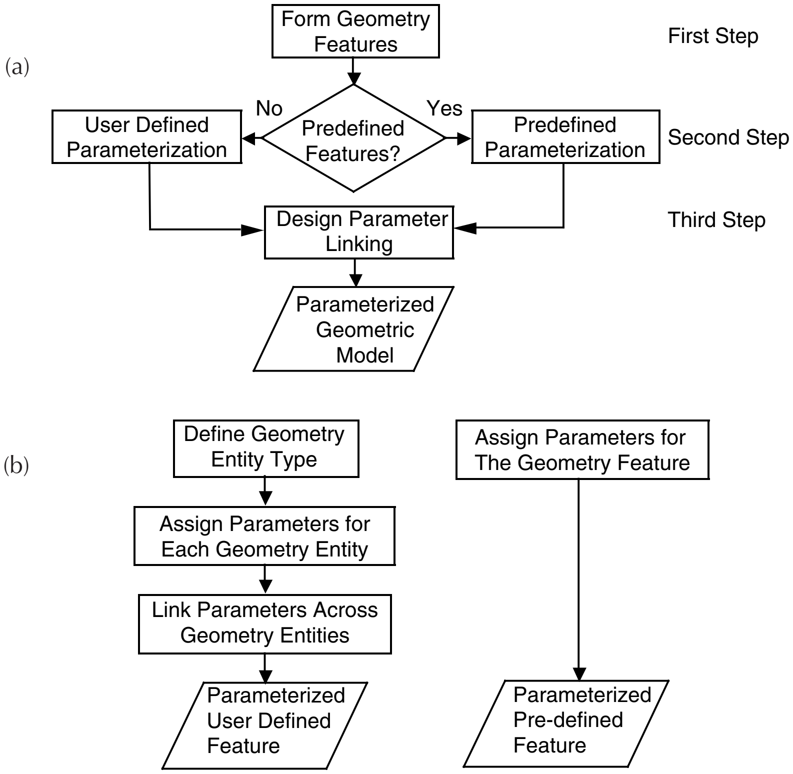

(a)
Form Geometry Features
First Step
No
Yes
User Defined Parameterization
Predefined Features?
Predefined Parameterization
Second Step
Design Parameter Linking
Third Step
Parameterized Geometric Model
(b)
Define Geometry Entity Type
Assign Parameters for Each Geometry Entity
Link Parameters Across Geometry Entities
Parameterized User Defined Feature
Assign Parameters for The Geometry Feature
Parameterized Pre-defined Feature

FIGURE 11.5.11 Shape design parameterization of features (a) overall design parameterization process; (b) parameterization for a user-defined feature; (c) parameterization for a predefined feature.

Therefore, one curve can be represented using different methods. However, these representations of curves and patches are mathematically equivalent, and one can be transformed into another by using certain linear transformations. For example, the geometric coefficients of a curve can be transformed into a four-point format, so the shape of the curve can be controlled according to the position of the four points. In fact, geometric coefficients are selected as unified geometric data, independent of the methods used to create geometric entities. Because design parameterization has the versatility of representing the same geometric entity in different ways, it can be systematically developed to provide the design engineer with sufficient resources for solving a wide variety of shape design problems.

The design parameterization method presented in this section is a three-step process, as illustrated in Figure 11.5.11. The first step is to create a geometric feature by grouping a number of interconnected geometric entities together, and by defining the type of geometric feature. The design engineer identifies the geometric entities that form the geometric features.

The second step is to define the design variables within each geometric feature. Geometric features that are frequently used in the construction of structural components can be put in the library of predefined geometric features. The design engineer can then parameterize these predefined features simply by selecting associated predefined shape design variables. To demonstrate the use of the library, two predefined geometric features have been defined, a circular hole and a tapered slot, as shown in Figure 11.5.12. For the circular hole, which is formed by connecting a number of circular arcs end to end with the same center point and radius, both the radius and center point of the circle can be defined as shape design variables. For the tapered slot, which is formed by connecting a number of straight lines and circular arcs, length dp3, radii dp4 and dp5, and center point dp1 and dp2 can all be defined as shape design variables. The design parameterization flow for the predefined geometric feature is illustrated in Figure 11.5.11(c).

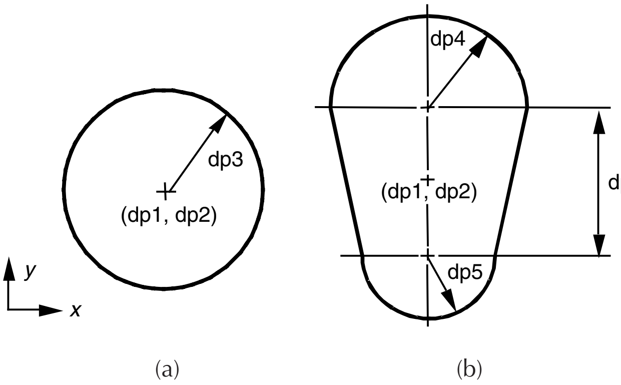


FIGURE 11.5.12 Predefined geometric features (a) circular hole; (b) tapered slot.

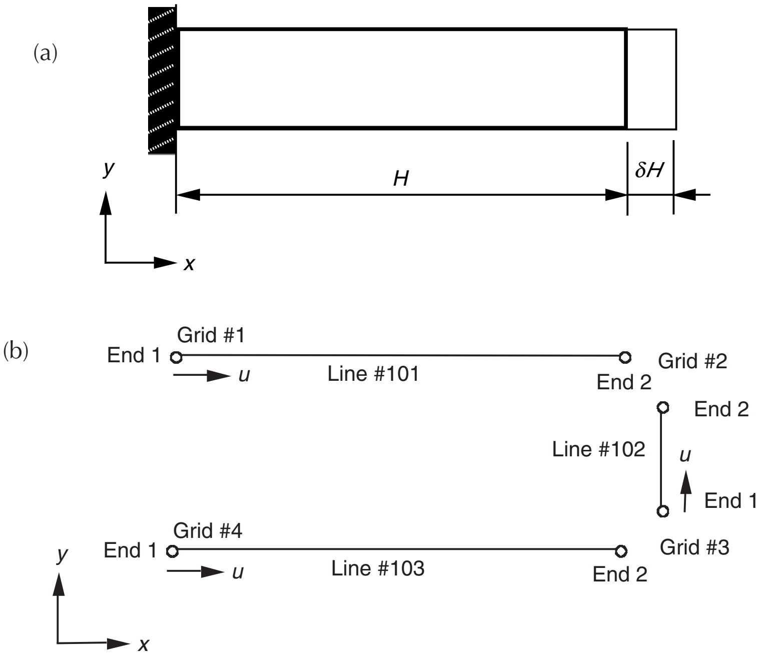


FIGURE 11.5.13 Parameterization of a cantilever beam (a) cantilever beam; (b) geometric feature of the beam.

A geometric feature that is not included in the library can be seen as a user-defined feature. To generate the latter, the design engineer can define the design variable by using geometric entities, and can then link these variables across the entities. For example, suppose a cantilever beam is to be parameterized such that its length *H* can be varied, as shown in Figure 11.5.13(a). To parameterize the beam, the following procedure can be used:

1. Identify lines #101, #102, and #103 to form the geometric feature, that is, the edges of the beam, as shown in Figure 11.5.13(b).
2. Define line #102 as a straight line and define the *x*-coordinate at end 1 of line 102, that is, grid #3, as the free design variable dp1, and the *x*-coordinate at end 2 of line #102, i.e., grid #2, to be proportional to dp1 with a proportionality of 1.0.
3. Define lines #101 and #103 as straight lines, and define the *x*-coordinate at end 2 of lines #101 and #103, that is, grids #2 and #3, as the free design variables dp2 and dp3, respectively.
4. Link dp2 and dp3 to dp1, with a proportionality of 1.0.

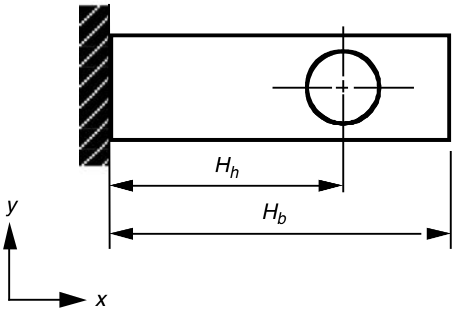


FIGURE 11.5.14 Design variable linking across parameterized geometric features.

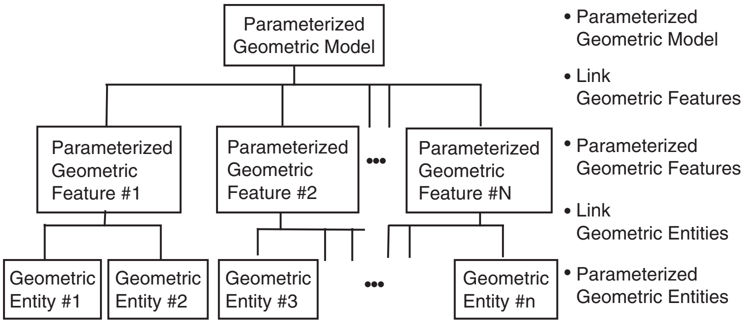

Parameterized Geometric Model
Parameterized Geometric Feature #1
Parameterized Geometric Feature #2
Parameterized Geometric Feature #N
Geometric Entity #1
Geometric Entity #2
Geometric Entity #3
Geometric Entity #n
• Parameterized Geometric Model
• Link Geometric Features
• Parameterized Geometric Features
• Link Geometric Entities
• Parameterized Geometric Entities

FIGURE 11.5.15 Hierarchy of shape design parameterization.

After design variable linking, only one design variable, dp1, the beam length represented by the *x*-coordinate of grid #3 in line #102, is allowed to vary independently. The parameterization flow for the user-defined feature is shown in Figure 11.5.11(b).

The shape design parameterization procedure for a user-defined geometric feature, as illustrated above, is summarized as follows:

1. Identify the types of geometric entities that are to be used to construct the user-defined geometric feature.
2. Parameterize the geometric entities by defining free and proportional design variables in each geometric entity.
3. Generate the parameterized geometric feature by linking free design variables across geometric entities.

Each predefined geometric feature that can be included in the feature library, such as the circular hole and tapered slot, is preconstructed by using this procedure.

If necessary, the third step is designed to link design variables across geometric features. For example, a cantilever beam with a circular hole, as shown in Figure 11.5.14, is to be parameterized so that the position of the hole is proportional to the beam length. The *x*-coordinate of the circular hole H_h can be parameterized by using the predefined parameterization process, as described in Figure 11.5.12(a). The length of beam H_b can be parameterized as a user-defined feature, as illustrated in Figure 11.5.13. With the two parameterized features, the *x*-coordinate of the hole can be linked to the beam length.

As described here, fundamental shape design parameterization is defined within geometric entities, and the parameterized features are created by using geometric entities. A hierarchy of the design parameterization method to build a parameterized geometric model is shown in Figure 11.5.15.

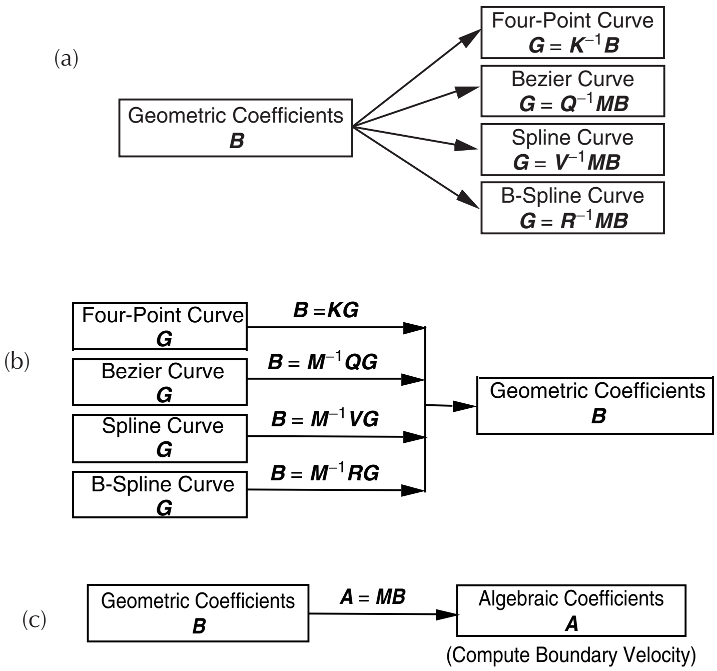


FIGURE 11.5.16 Curve format transformations for two-dimensional shape design (a) transformations from **B** to **G**; (b) transformations from **G** to **B**; (c) transformations from **B** to **A**.

Curve Design Parameterization

For a two-dimensional shape design, the boundaries are planar curves. In general, there are eight degrees of freedom for a planar cubic curve, as expressed in Equation 11.5.1 and Equation 11.5.2, with $z(u)$ serving as the constant. Planar curves with eight degrees of freedom are designated as basic curves, while predefined curves that are constrained, such as a circular arc, are designated as specialized curves. A specialized curve has fewer degrees of freedom since some of the basic degrees of freedom are linked (constrained) in order to define the required characteristics of the curve.

From a computational point of view, algebraic and geometric curves are the most interesting among the six basic types of curves. All parametric cubic entities can be transformed into various other formats by using certain linear transformations. For shape design, three major transformations are necessary: (1) from the geometric coefficient matrix **B** to the design variable matrix **G** to compute design variable values; (2) from matrix **G** to matrix **B** to update geometric entities for a perturbed design shape; and (3) from matrix **G** to the algebraic coefficient matrix **A** to compute the boundary velocity field. For the basic curves, the transformation from matrix **G** to matrix **B** for each curve format can be described by their corresponding 4 × 4 constant matrices. The curve format transformations are summarized in Figure 11.5.16.

Surface Design Parameterization

For three-dimensional shape design, design boundaries are surfaces in space. In general, there are 48 degrees of freedom for a parametric bicubic surface. For a parametric bicubic surface, the x-, y-, and z-components can be expressed using the three functions, as

$$\left.\begin{aligned} x &= x(u,w) \\ y &= y(u,w) \\ z &= z(u,w) \end{aligned}\right\} \tag{11.5.7}$$

where u and w are the parametric directions of the geometric entity, and $(u, w) \in [0,1] \times [0,1]$. Each of these functions is, at most, a bicubic function of the form

$$\begin{aligned} z(u,w) &= a_{33}u^3w^3 + a_{32}u^3w^2 + a_{31}u^3w + a_{30}u^3 \\ &\quad + a_{23}u^2w^3 + a_{22}u^2w^2 + a_{21}u^2w + a_{20}u^2 \\ &\quad + a_{13}u^1w^3 + a_{12}u^1w^2 + a_{11}u^1w^1 + a_{10}u \\ &\quad + a_{03}w^3 + a_{02}w^2 + a_{01}w + a_{00} \\ &= \sum_{i,j=0}^{3} a_{ij}u^iw^j \end{aligned} \tag{11.5.8}$$

Surfaces with 48 degrees of freedom are defined as basic surfaces, whereas specialized surfaces are constrained to represent predefined conditions. As with the specialized curve, the specialized surface has fewer degrees of freedom. There are four basic surfaces — algebraic, geometric, 16-point, and Bezier — and there are also four specialized surfaces — plane, cylindrical, ruled, and surface of revolution, which have been developed to handle three-dimensional shape design problems. The B-spline surface is not recommended for three-dimensional shape design since, in contrast to the B-spline curve, the control points at the edges of the polyhedron are not on the B-spline surface. Therefore, the physical surface boundary does not closely resemble the characteristic polyhedron, and consequently, it is difficult to use in geometric modeling and design.

Similar to the basic types of curves, transformations between matrices **G**, **A**, and **B** of the basic surface can be used to obtain shape design variables, to update geometric entities, and to compute the boundary velocity field. The various surface format transformations are categorized in Figure 11.5.17.

The Configuration Design Variable

For those built-up structures made of truss, beam, and shell components, there is another type of design variable in addition to shape design; it's called the configuration design variable, and it is related to the structural component's orientation. These components have local coordinate systems fixed on the body of the structure, and state variables of the problem are described in local coordinate systems. If several components are connected together for the built-up structure, the state variables described in the local coordinate system are transformed to the global coordinate system. If the structural components change

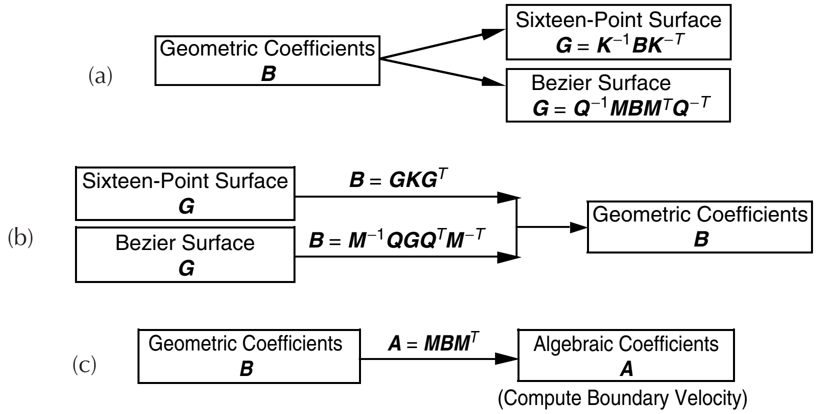

(a)
Geometric Coefficients $\boldsymbol{B}$
Sixteen-Point Surface $\boldsymbol{G} = \boldsymbol{K}^{-1}\boldsymbol{B}\boldsymbol{K}^{-T}$
Bezier Surface $\boldsymbol{G} = \boldsymbol{Q}^{-1}\boldsymbol{M}\boldsymbol{B}\boldsymbol{M}^{T}\boldsymbol{Q}^{-T}$
(b)
Sixteen-Point Surface $\boldsymbol{G}$
Bezier Surface $\boldsymbol{G}$
$\boldsymbol{B} = \boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^{T}$
$\boldsymbol{B} = \boldsymbol{M}^{-1}\boldsymbol{Q}\boldsymbol{G}\boldsymbol{Q}^{T}\boldsymbol{M}^{-T}$
Geometric Coefficients $\boldsymbol{B}$
(c)
Geometric Coefficients $\boldsymbol{B}$
$\boldsymbol{A} = \boldsymbol{M}\boldsymbol{B}\boldsymbol{M}^{T}$
Algebraic Coefficients $\boldsymbol{A}$
(Compute Boundary Velocity)

FIGURE 11.5.17 Surface format transformations for three-dimensional shape design (a) transformations from **B** to **G**; (b) transformations from **G** to **B**; (c) transformations from **B** to **A**.

their orientation in space, the transformation between the local and global coordinates also changes. Thus, this transformation can be considered the configuration design variable. Since configuration design variables are defined for built-up structures, they are inherently coupled with shape design variables. That is, in order to allow one member of the built-up structure to rotate, another member's shape needs to be changed. The configuration design variable is not applicable to solid components in which all rotations can be expressed in terms of shape changes.

The Topology Design Variable

If shape and configuration design variables represent changes in structural geometry and orientation, then topology design determines the structure's layout. For example, in Figure 11.5.9, shape design can change the size and location of the slot within the block. However, shape design cannot completely remove the slot from the block, or introduce a new slot. Topology design determines whether the slot can be removed or an additional slot is required.

The choice of the topology design variable is nontrivial compared to other design variables. Which parameter is capable of representing the birth or death of the structural layout? Early developments in topology design focused on truss structures. For a given set of points in space, design engineers tried to connect these points using truss structures, in order to find the best layout to support the largest load. Thus, the on-off types of topology design variables are used. These kinds of designs, however, could turn out to be discontinuous and unstable.

Recent developments in topology design are strongly related to FEA. The candidate design domain is modeled using finite elements, and then the material property of each element is controlled. If it is necessary to remove a certain region, then the material property value (e.g., Young's modulus) will approach zero, such that there will be no structural contribution from the removed region. Thus, material property design variables could be used for the purpose of topology design variables. The on-off type of design variable can be approximated by using continuous polynomials in order to remove the difficulties associated with discrete design variables.

In many applications, topology design is used at the concept design stage such that the layout of the structure is determined. After the layout is determined, sizing and shape designs are used to determine the detailed geometry of the structure.

A final comment on design parameterization: it is desirable to have a linearly independent set of design variables. If one does not, then relations between design variables must be imposed as constraints, which may make the design optimization process expensive, as the number of design variables and constraints increase. Furthermore, if design variable constraints are not properly established, meaningless design results will be obtained after an extensive amount of computational effort. As mentioned before, this problem is strongly related to structural modeling, since a well-defined structural model should have an independent set of parameters to define the entire system. Even if defining a good model is not an easy task for a complicated design problem, the design engineer nevertheless has to define a proper and independent set of parameters as much as possible in the structural modeling stage.

Design Variable Linking across Geometric Entities

There are three methods for categorizing the shape design problem: (1) identifying the arbitrary boundary shape, (2) determining the dimensions of the predefined shape, and (3) finding the locations of the boundaries. A design variable linking process is discussed in this section in order to support and ensure continuity between boundaries, as well as to retain predefined boundary shapes. For a geometric feature that is formed by a set of B-spline curves, C^2-continuity across curves is naturally retained.

As described in the previous sections, for the first type of shape design boundary any basic curve or surface can be utilized for parameterization. If constrained boundary shapes are required for two-dimensional structures, such as a straight edge or circular arc, then that specific type of curve must be used. For three-dimensional structures, the constrained boundaries can be predefined, such as a plane, cylinder, ball, or surface of revolution. If the geometric feature is composed of more than one curve or surface, continuity across curves or surfaces must be retained by linking shape design variables at the joint

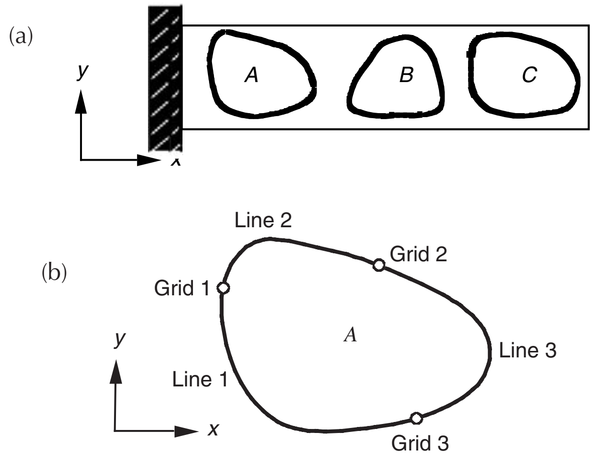


FIGURE 11.5.18 Parameterization of arbitrarily shaped holes (a) beam with arbitrarily shaped holes; (b) Hole *A* formed by three curves.

point or curve. As mentioned before, C^0-, C^1-, and C^2-continuities can be retained for two-dimensional structures by using the appropriate curve type and design variable linking. For three-dimensional problems, C^2-continuity is difficult to maintain. The following three examples illustrate how the design variable linking process can be used to support the three types of shape design problems.

Type One Problems — Identifying the Arbitrary Boundary Shape

A two-dimensional beam with arbitrarily shaped holes, shown in Figure 11.5.18(a), illustrates a design variable linking process that supports the first type of shape design problem.

Figure 11.5.18(b) shows that hole *A* is formed by three curves. To parameterize this hole, the curves are defined as geometric curves. The endpoints and tangent vectors at the endpoints of these connected curves are linked to retain C^0- and C^1-continuities, respectively.

The shape of the hole can be changed due to endpoint movement, length, or the direction of the tangent vector, as shown in Figure 11.5.19. Hole shape change due to movement δdp_1 of grid 3 in the negative *y*-direction is shown in Figure 11.5.19(a), such that the tangent vectors of lines 1 and 3 at grid 3 are kept the same. Moreover, hole shape change due to scaling the tangent vector of line 3 at grid 3 by δdp_2, shown in Figure 11.5.19(b), is such that the location of grid 3 and the direction of the tangent vector are kept the same. Finally, hole shape change due to change δdp_3 in the tangent vector direction of lines 1 and 3 is shown in Figure 11.5.19(c).

To avoid meaningless designs, for example, a hole that penetrates the boundary edges of a beam, the geometric boundaries can be displayed after the design change. Furthermore, numerical limits can be defined for design variables in order to restrict the degree of design perturbation.

Type Two Problems — Determining the Dimensions of a Predefined Shape

For the second type of shape design problem, specialized curves and surfaces are used to parameterize geometric entities. In addition, design variable linking may be carried out in the design process to group curves and surfaces that define the geometric feature.

The design variable linking method for parameterizing the second type of shape design boundary can be described using an elliptic hole formed by four conic curves, as shown in Figure 11.5.20. Each conic curve has three points to control its shape; however, relative altitude ρ is kept constant.

To vary the major axis, the *x*-coordinate of point $\mathbf{p}_7$ is defined as the independent design variable dp1. Points $\mathbf{p}_6$ and $\mathbf{p}_8$ are linked to dp1, with a proportionality of 1.0. In addition, the *x*-coordinates of points 1, 2, and 3 are linked to dp1 with a proportionality of –1.0. The elliptic hole varies in shape when the major axis is perturbed by δdp1, as shown in Figure 11.5.21(a). Similarly, for a design change in the minor axis, the *y*-coordinate of point $\mathbf{p}_4$ is defined as the independent design variable dp2. Points $\mathbf{p}_1$ and

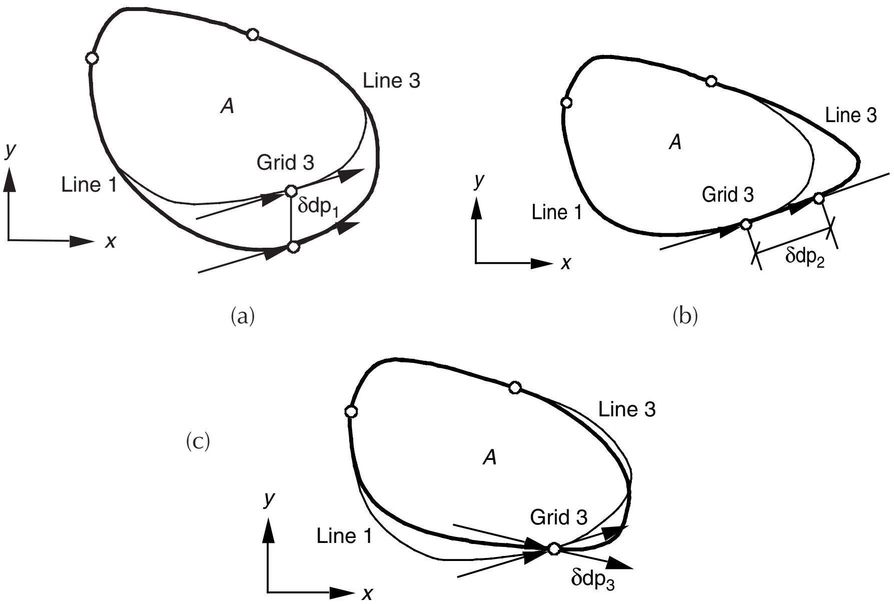


FIGURE 11.5.19 Shape variation of Hole A (a) grid point movement; (b) length of tangent vector change; (c) *y*-component of tangent vector change.

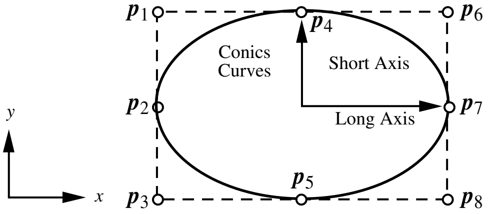


FIGURE 11.5.20 Parameterization of an elliptic hole.

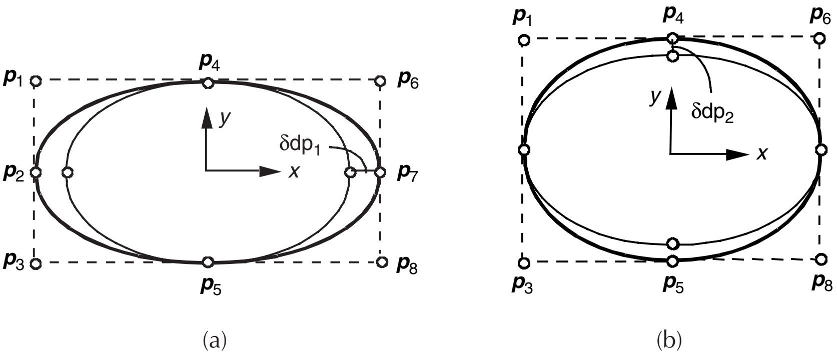


FIGURE 11.5.21 Shape variation of an elliptic hole (a) variation of major axis; (b) variation of minor axis.

$\mathbf{p}_6$ are linked to dp2, with a proportionality of 1.0. Also, the *y*-coordinates of points $\mathbf{p}_3$, $\mathbf{p}_5$, and $\mathbf{p}_8$ are linked to dp2, with a proportionality of –1.0. Shape change due to the perturbation δdp2 of the minor axis is shown in Figure 11.5.21(b).

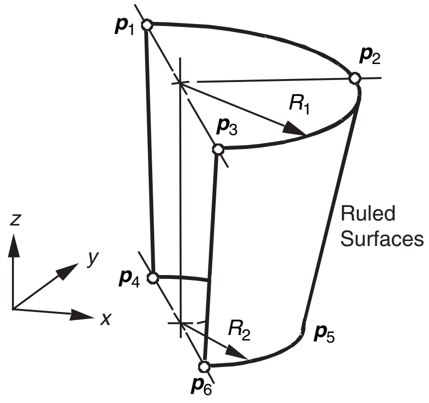


FIGURE 11.5.22 A semitapered cylinder formed by two ruled surfaces.

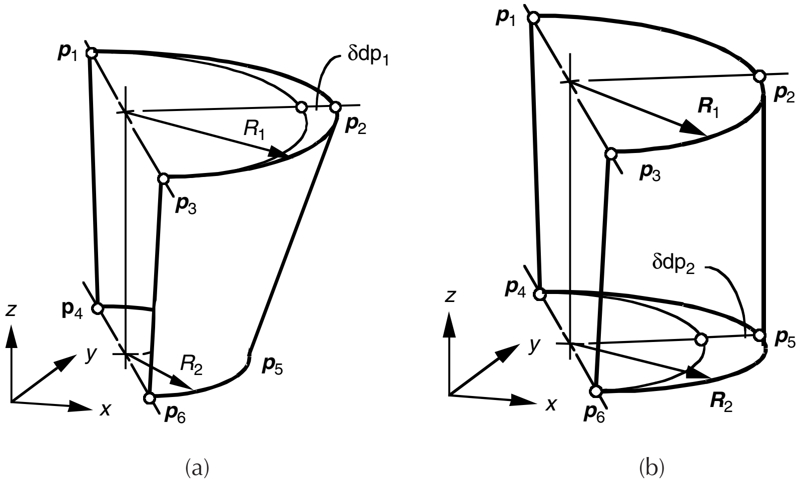


FIGURE 11.5.23 Shape variation of a tapered semicylindrical surface (a) shape variation due to δdp1; (b) shape variation due to δdp2.

For three-dimensional problems, this type of shape design can be the radius of a ball, a shell, or a cylindrical hole. For example, the tapered semicylindrical surface shown in Figure 11.5.22 can be parameterized by linking the design variables defined in the two ruled surfaces that form the tapered semicylindrical surface. The first ruled surface $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_4\mathbf{p}_5$ is created using circular arcs $\mathbf{p}_1\mathbf{p}_2$ and $\mathbf{p}_4\mathbf{p}_5$. The other ruled surface $\mathbf{p}_2\mathbf{p}_3\mathbf{p}_5\mathbf{p}_6$ is created using circular arcs $\mathbf{p}_2\mathbf{p}_3$ and $\mathbf{p}_5\mathbf{p}_6$.

To vary the radius R_1, the radius of arc $\mathbf{p}_1\mathbf{p}_2$ is defined as the independent design variable dp1, and the radius of arc $\mathbf{p}_2\mathbf{p}_3$ is linked to dp1 with a proportionality of 1.0. Similarly, the design variable dp2 can be defined for radius R_2. The shape of the tapered semicylindrical surface is defined by the shape design variables dp1 and dp2, as shown in Figure 11.5.23(a) and Figure 11.5.23(b), respectively.

Type Three Problems — Finding the Locations of the Boundaries

To support the third type of shape design problem, such geometric entities as curves or surfaces that form the geometric feature are linked together so that only *x*-, *y*-, and *z*-movements of the parameterized geometric feature are allowed. For example, the *x*-coordinate of points $\mathbf{p}_1$ to $\mathbf{p}_8$ of the elliptic hole shown in Figure 11.5.22 can be linked, with a proportionality of 1.0, so that the four conic curves can be moved together to form an elliptic hole.

Design Sensitivity Analysis

Design sensitivity analysis computes the rate of performance measure change with respect to design variable changes. With the structural analysis, the design sensitivity analysis generates a critical information, gradient, for design optimization. Obviously, the performance measure is presumed to be a differentiable function of the design, at least in the neighborhood of the current design point. For complex engineering applications, it is not simple to prove a performance measure's differentiability with respect to the design. For most problems in this section, one can assume that the performance measure is continuously differentiable with respect to the design.

In general, a structural performance measure depends on the design. For example, a change in the cross-sectional area of a beam would affect the structural weight. This type of dependence is simple if the expression of weight in terms of the design variables is known. For example, the weight of a straight beam with a circular cross-section can be expressed as

$$W(r) = \pi r^2 l$$

where $u = r$ is the radius and l is the length of the beam. If the radius is a design variable, then the design sensitivity of W with respect to r would be

$$\frac{dW}{dr} = 2\pi r l$$

This type of function is *explicitly dependent* on the design, since the function can be explicitly written in terms of that design. Consequently, only algebraic manipulation is involved, and no FEA is required to obtain the design sensitivity of an explicitly dependent performance measure.

However, in most cases, a structural performance measure does not explicitly depend on the design. For example, when the stress of a beam is considered as a performance measure, there is no simple way to express the design sensitivity of stress explicitly in terms of the design variable r. In the linear elastic problem, the stress of the structure is determined from the displacement, which is a solution to the FEA. Thus, the sensitivity of stress $\sigma(\mathbf{z})$ can be written as

$$\frac{d\sigma}{dr} = \frac{d\sigma}{d\mathbf{z}}^{\mathrm{T}} \frac{d\mathbf{z}}{dr} \tag{11.5.9}$$

where $\mathbf{z}$ is the displacement of the beam. Since the expression of stress as a function of displacement is known, $d\sigma/d\mathbf{z}$ can be easily obtained. The only difficulty is the computation of $d\mathbf{z}/dr$, which is the state variable (displacement) sensitivity with respect to the design variable r.

When a design engineer wants to compute the design sensitivity of performance measures such as stress $\sigma(\mathbf{z})$ in Equation 11.5.9, structural analysis (FEA, for example) has presumably already been carried out. Assume that the structural problem is governed by the following linear algebraic equation:

$$\mathbf{K}(u)\mathbf{z} = \mathbf{f}(u) \tag{11.5.10}$$

Equation 11.5.10 is a matrix equation of finite elements if $\mathbf{K}$ and $\mathbf{f}$ are understood to be the stiffness matrix and load vector, respectively. Suppose the explicit expressions of $\mathbf{K}(u)$ and $\mathbf{f}(u)$ are known and differentiable with respect to u. Since the stiffness matrix $\mathbf{K}(u)$ and load vector $\mathbf{f}(u)$ depend on the design u, solution $\mathbf{z}$ also depends on the design u. However, it is important to note that this dependency is implicit, which is why we need to develop a design sensitivity analysis methodology. As shown in Equation 11.5.9, $d\mathbf{z}/du$ must be computed using the governing equation of Equation 11.5.10. This can be achieved by differentiating Equation 11.5.10 with respect to u, as

$$\mathbf{K}(u)\frac{d\mathbf{z}}{du} = \frac{d\mathbf{f}}{du} - \frac{d\mathbf{K}}{du}\mathbf{z} \tag{11.5.11}$$

Assuming that the explicit expressions of $\mathbf{K}(u)$ and $\mathbf{f}(u)$ are known, $d\mathbf{K}/du$ and $d\mathbf{f}/du$ can be evaluated. Thus, if solution $\mathbf{z}$ in Equation 11.5.10 is known, then $d\mathbf{z}/du$ can be computed from Equation 11.5.11, which can then be substituted into Equation 11.5.9 to compute $d\sigma/du$. Note that the stress performance measure is *implicitly dependent* on the design through state variable $\mathbf{z}$.

In this text, it is assumed that the general performance measure ψ depends on the design explicitly and implicitly. That is, the performance measure ψ is presumed to be a function of design u, and state variable $\mathbf{z}(u)$, as

$$\psi = \psi\big(\mathbf{z}(u),u\big) \tag{11.5.12}$$

The sensitivity of ψ can thus be expressed as

$$\frac{d\psi(\mathbf{z}(u),u)}{du} = \left.\frac{\partial\psi}{\partial u}\right|_{\mathbf{z}=\text{const}} + \left.\frac{\partial\psi}{\partial \mathbf{z}}\right|_{u=\text{const}}^{T} \frac{d\mathbf{z}}{du} \tag{11.5.13}$$

The only unknown term in Equation 11.5.13 is $d\mathbf{z}/du$. Various computational methods to obtain $d\mathbf{z}/du$ are introduced in the following subsections.

Methods of Structural Design Sensitivity Analysis

Various methods employed in design sensitivity analysis are listed in Figure 11.5.24. Four approaches are used to obtain the design sensitivity: the finite difference, discrete, continuum, and computational derivatives. In the finite difference approach, design sensitivity is obtained by either the *forward finite difference* or the *central finite difference method*. In the discrete method, design sensitivity is obtained by taking design derivatives of the discrete governing equation. For this process, it is necessary to take the design derivative of the stiffness matrix. If this derivative is obtained analytically using the explicit expression of the stiffness matrix with respect to the design variable, it is an *analytical method*, since the analytical expressions of $\mathbf{K}(u)$ and $\mathbf{f}(u)$ are used. However, if the derivative is obtained using a finite difference method, the method is called a *semianalytical method*. In the continuum approach, the design derivative of the variational equation is taken before it is discretized. If the structural problem and sensitivity equations are solved as a continuum problem, then it is called the *continuum-continuum method*. However, only very simple, classical problems can be solved analytically. Thus, the continuum sensitivity equation is solved by discretization in the same way that structural problems are solved. Since differentiation is taken at the continuum domain and is then followed by discretization, this method is

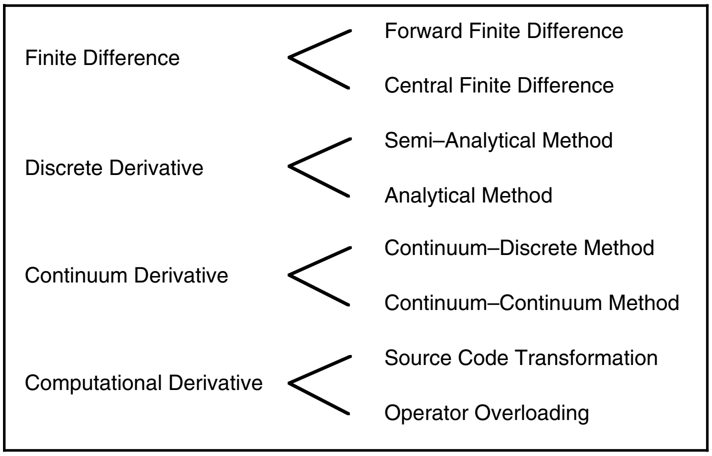


FIGURE 11.5.24 Approaches to design sensitivity analysis.

called the *continuum-discrete method*. Finally, computational, algorithmic or automatic differentiation refers to a differentiation of the computer code itself.

Except for the finite differences option, the other three come in direct and adjoint methods (called the reverse mode for computational derivative). In the direct method, one obtains the derivatives of the entire structural response, and often of intermediate quantities as well. The sensitivities of performance measures can then be obtained from the chain rule of differentiation. In the adjoint method, one defines an adjoint problem, which depends on the performance measure. The sensitivities of performance measures can then be obtained using the structural and adjoint responses. Thus, all of the system response sensitivities are not required, which is particularly an advantage in cases with many design variables, though few performance measures of interest.

The Global Finite Difference Method

The easiest way to compute sensitivity information of the performance measure is by using the finite difference method. Different designs yield different analysis results and, thus, different performance values. The finite difference method actually computes design sensitivity of performance by evaluating performance measures at different stages in the design process. If u is the current design, then the analysis results provide the value of performance measure $\psi(u)$. In addition, if the design is perturbed to $u+\Delta u$, where Δu represents a small change in the design, then the sensitivity of $\psi(u)$ can be approximated as

$$\frac{d\psi}{du} \approx \frac{\psi(u+\Delta u)-\psi(u)}{\Delta u} \tag{11.5.14}$$

Equation 11.5.14 is called the *forward difference method* since the design is perturbed in the direction of $+\Delta u$. If $-\Delta u$ is substituted in Equation 11.5.14 for Δu, then the equation is defined as the *backward difference method*. Additionally, if the design is perturbed in both directions, such that the design sensitivity is approximated by

$$\frac{d\psi}{du} \approx \frac{\psi(u+\Delta u)-\psi(u-\Delta u)}{2\Delta u} \tag{11.5.15}$$

then the equation is defined as the *central difference method*.

The advantage of the finite difference method is obvious. If structural analysis can be performed and the performance measure can be obtained as a result of structural analysis, then the expressions in Equation 11.5.14 and Equation 11.5.15 are virtually independent of the problem types considered. Consequently, this method is still popular in engineering design.

However, sensitivity computation costs become the dominant concern in the design process. If n represents the number of designs, then $n+1$ number of analyses have to be carried out for either forward or backward difference method, and $2n+1$ analyses are required for the central difference method. For modern, practical engineering applications, the cost of structural analysis is rather expensive. Thus, this method is not feasible for large-scale problems containing many design variables.

Another major disadvantage of the finite difference method is the accuracy of its sensitivity results. In Equation 11.5.14, accurate results can be expected when Δu approaches zero. Figure 11.5.25 shows some sensitivity results using the finite difference method. The tangential slope of the curve at u_0 is the exact sensitivity value. Depending on perturbation size, we can see that sensitivity results are quite different. For a mildly nonlinear performance measure, relatively large perturbation provides a reasonable estimation of sensitivity results. However, for highly nonlinear performances, a large perturbation yields completely inaccurate results. Thus, the determination of perturbation size greatly affects the sensitivity result. And even though it may be necessary to choose a very small perturbation, numerical noise becomes dominant for a too small perturbation size. That is, with a too small perturbation, no reliable difference can be found in the analysis results. The most obvious source of this type of errors is computational errors associated with arithmetic with finite number of digits and possibly ill-conditioning in the problem.

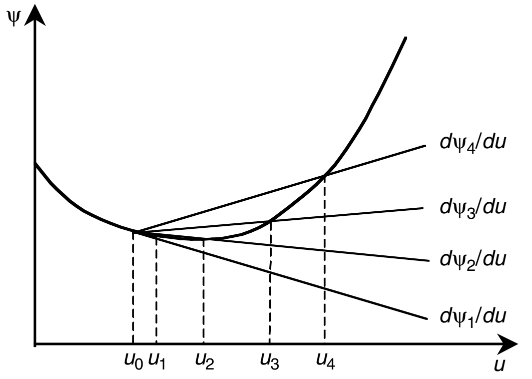


FIGURE 11.5.25 Influence of step-size in forward finite difference method.

For example, if up to five digits of significant numbers are valid in a structural analysis, then any design perturbation in the finite difference that is smaller than the first five significant digits cannot provide meaningful results. As a result, it is difficult to determine design perturbation sizes that work for all problems. Other potential sources are the discretization of both the spatial and the temporal domain. A typical example could be numerical noise induced by re-meshing.

Computational efficiency, accuracy and consistency, and implementation effort for global finite differences depend to a large extent on the type of solvers used for the linear system of Equation 11.5.10. The main issue is whether computational investments associated with solving the equations for the nominal structure can help reduce the effort associated with solving these equations for a perturbed structure.

When direct solvers are used for the solution, so that the matrix **K** has been factored, there is an array of methods that provide fast re-analysis of the perturbed structure. A disadvantage of many of these techniques is that accuracy is generally compromised, that is, certain inaccuracies will be introduced. When the perturbation leads to a low-rank modification of **K** — for example, because only a single finite element is modified — then an exact analysis of the perturbed structure can be performed using the Sherman-Morrison-Woodbury formulas.[19] The main computational cost of this approach is the solution of Equation 11.5.10 for a number of right-hand sides equal to the rank of the perturbation in **K**. Akgün et al.[19] discuss several variants of this approach, including the method of virtual distortions. When the perturbation in the matrix is more extensive, as in shape variation, it is still possible to use a binomial series solution[20,21] or a similar approximation of the inverse of **K** using a Neuman series.[22,23]

The Discrete Method

A structural problem is often discretized in finite dimensional space in order to solve complex problems. The discrete method computes the performance design sensitivity of the discretized problem, where the governing equation is a system of linear equations, as in Equation 11.5.10. If the explicit form of the stiffness matrix **K**(u) and the load vector **f**(u) are known, and if solution **z** of matrix equation **K**(u)**z** = **f**(u) is obtained, then the design sensitivity of the displacement vector can also be obtained, from Equation 11.5.11, as

$$\mathrm{K}(u)\frac{d\mathbf{z}}{du} = \mathbf{p} \tag{11.5.16}$$

where the pseudo-load vector **p** is defined as

$$\mathbf{p} = \frac{d\mathrm{f}}{du} - \frac{d\mathrm{K}}{du}\mathbf{z} \tag{11.5.17}$$

It is clear from Equation 11.5.16 that the design sensitivities require the solution of the same set of equations as solved for the response functions, but for another right-hand side (compare with Equation 11.5.10), the latter being the pseudo-load vector (see Equation 11.5.17).

In calculating the pseudo-load vector, it is not necessary to differentiate the global load vector and stiffness matrix, but to differentiate only those elements that are affected by the design variable. The evaluation of the pseudo-load vector is then carried out by an assembly of all individual nodal points and finite element contributions. These contributions are obtained by differentiating the finite element stiffness matrices with respect to the design variables and following a similar procedure for all load contributions. The fact that the pseudo-load vector only depends on elements that are affected may be exploited to make the computation of the pseudo-load vector more efficient. For shape design variables, this requires some additional attention. For that purpose, one often tries to link the design variables only to boundary elements, which means that only a boundary layer of elements is affected by the shape design variables.

The analytical differentiation process may become tedious. This especially holds true for shape design variables. Additional procedures must be implemented for each element used within the sensitivity analysis. The procedure must account for all possible design variables and particularly for shape design variables, as they are usually more complex than the original finite element routines. This type of discrete design sensitivity is referred to as *analytical* discrete design sensitivity.

It is not difficult to compute $d\mathbf{f}/du$, since the applied force is usually either independent of the design, or it has a simple expression. However, the computation of $d\mathbf{K}/du$ in Equation 11.5.16 depends on the type of problem. In addition, modern advances in the finite element method use numerical integration in the computation of $\mathbf{K}$. In this case, the explicit expression of $\mathbf{K}$ in terms of u may not be available. Moreover, in the case of the shape design variable, computation of the analytical derivative of the stiffness matrix is quite costly. Because of this, frequently approximations are accepted for the pseudo-load vector that reduces this effort. These approximations particularly involve finite difference schemes for evaluation of the pseudo-load vector. Forward and central finite difference schemes are the most popular. This type of design sensitivity is commonly denoted *semianalytical* discrete design sensitivity. However, Barthelemy and Haftka[24] show that the semianalytical method can have serious accuracy problems for shape design variables in structures modeled by beam, plate, truss, frame, and solid elements. They found that accuracy problems occur even for a simple cantilever beam. Moreover, errors in the early stage of approximation multiply during the matrix equation solution phase. As a remedy, Olhoff et al.[25] proposed an exact numerical differentiation method when the analytical form of the element stiffness matrix is available.

For shape design variables, design perturbation involves both the size of the perturbation and its distribution over the domain. For the choice of perturbation size, similar considerations as discussed for global finite differences play a role. Unfortunately, the semianalytical formulation may be extremely sensitive with respect to this choice. We will come back to this aspect extensively, and we only note here that this drawback may negate all advantages of a semianalytical formulation and motivates modifications to the semianalytical method.

The Continuum Method

In the continuum method, the design derivative of the variational equation (the continuum model of the structure) is taken before discretization. Since differentiation is taken before any discretization takes place, this method provides more accurate results than the discrete approach. In addition, profound mathematical proofs are available regarding the existence and uniqueness of the design sensitivity.

Sizing design variables are distributed parameters of the continuum equation. For shape design variables, the material derivative concept of continuum mechanics is used to relate variations in structural shape to the structural performance measures.[4] Using the continuum approach, we can obtain design sensitivity expressions in the form of integrals, with integrands written in terms of such physical quantities as displacement, stress, strain, and domain shape change. If exact solutions to the continuum equations are used to evaluate these design sensitivity expressions, then this procedure is referred to as the continuum-continuum method. On the other hand, if approximation methods such as the finite element, boundary element, or mesh-free method are used to evaluate these terms, then this procedure is called the continuum-discrete method. The continuum-continuum method provides the exact design sensitivity of the exact model, whereas the continuum-discrete method provides an approximate design sensitivity of the exact model. When FEA is used to evaluate the structural response, then the same discretization method as structural analysis has to be used to compute the design sensitivity of performance measures in the continuum-discrete method.

In the continuum approach, the design variables may be considered as fields that are functions of the spatial coordinates. As a consequence, sensitivity is to be understood as a variation of a function. Let us consider that the design variable *s* is perturbed to *s* + τη in which τ is the scalar that measures the perturbation size and η is the direction of design change. For simplicity, it is assumed that the structural design variable *s* does not affect the domain. The variation of field response **u** with respect to *s* can then be defined as

$$\mathbf{u}' \equiv \lim_{\tau\to 0}\left\{\frac{\mathbf{u}(s+\tau\eta)-\mathbf{u}(s)}{\tau}\right\} = \left.\frac{\partial \mathbf{u}}{\partial \tau}\right|_{\tau=0}\eta \tag{11.5.18}$$

Since the direction of design change η can be arbitrary, Equation 11.5.18 must be linear with respect to η and the coefficient of η is called the sensitivity of field response **u**, which is equivalent to the derivative in the context of other approaches.

The continuum method of design sensitivity analysis starts from the principle of virtual work, which is convenient for formulating the equations of equilibrium, as

$$\iint_V \delta\varepsilon^T \mathbf{C}\varepsilon\, dV = \iint_{\delta \mathbf{u}^T \mathbf{b} dV} \delta\mathbf{u}^T \mathbf{b}\, dV + \int_A \delta\mathbf{u}^T \mathbf{h}\, dA \tag{11.5.19}$$

for all δ**u** that belong to the space of kinematically admissible displacements. In Equation 11.5.19, δε is the virtual strain; **C** is the elasticity matrix; ε is the strain vector; **b** denotes the external load per unit volume; and **h** reflects tractions acting on the outer surface *A* of the structure.

AU: What is this referring to?

Using Equation 11.5.18, the equations of equilibrium, (2.3), can be differentiated to obtain the following continuum sensitivity equation:

$$\iint_V \delta\varepsilon^T \mathbf{C}\varepsilon'\, dV = \iint_V \delta\mathbf{u}^T \mathbf{b}'\, dV + \int_A \delta\mathbf{u}^T \mathbf{h}'\, dA - \iint_V \delta\varepsilon^T \mathbf{C}'\varepsilon\, dV \tag{11.5.20}$$

for all δ**u** that belong to the space of kinematically admissible displacements. The left side of Equation 11.5.20 is the same as the left side of Equation 11.5.19 if **u** is replaced by **u**′. The right side of Equation 11.5.20 defines a pseudo-load (or fictitious load), which explicitly depends on the design. Thus, solving the sensitivity equation is the same as solving original structural equilibrium equation with different load terms. The major advantage of the continuum approach is that the sensitivity formulation is independent of discrete model and numerical schemes. Once the continuum sensitivity equation is obtained, it can be discretized in the same manner as the original analysis equations in order to obtain a system of matrix equations similar to Equation 11.5.10.

When the design variables affect the shape of the domain, the differentiation of the equations of equilibrium is much more complicated because the integral domain depends on the design. Interested readers are referred to Haug et al.[4] for the material derivative approach and Arora[26] or Phelan and Haber[27] for the control volume approach.

One frequently asked question is "Are the discrete and continuum-discrete methods equivalent?" To answer this question, we have to give four conditions. First, the same discretization (shape function) used in the FEA method must be used for continuum design sensitivity analysis. Second, an exact integration (instead of a numerical integration) must be used in the generation of the stiffness matrix and in the evaluation of continuum-based design sensitivity expressions. Third, the exact solution (and not a numerical solution) of the finite element matrix equation and the adjoint equation should be used to compare these two methods. Fourth, the movement of discrete grid points must be consistent with the design parameterization method used in the continuum method. For the sizing design variable, it is shown in reference 4 that the discrete and continuum-discrete methods are equivalent under the conditions given above, using a beam as the structural component. It has also been argued that the discrete and continuum-discrete methods are equivalent in shape design problems under the conditions given above.[28] One point to note is that these four conditions are not easy to satisfy; in many cases, numerical integration is used and exact solutions of the FE matrix equation cannot be obtained.

Computational Derivative

Even if the finite element programs are composed of many complicated subroutines and functions, they are basically a collection of elementary functions. Computational (or automatic) differentiation method defines the partial derivatives of these elementary functions, and then the derivatives of complicated subroutines and functions are computed using propagation and the chain rule of differentiation. The arguments of elementary functions can be either one or two. Without loss of generality, let us assume that an elementary function has two arguments, defined as

$$a = f_{\text{elem}}(z_i, z_j) \tag{11.5.21}$$

where $f_{\text{elem}}(\ ,\ )$ represents $(\ {+}{+}, \sin(\ ), \ldots\ )$ operators for the single argument and (+, –, *, /, …) operators for the double arguments.

In the direct differentiation method, the derivative of Equation 11.5.21 can be defined as

$$\frac{\partial a}{\partial s} = \frac{\partial f_{\text{elem}}}{\partial z_i}\frac{\partial z_i}{\partial s} + \frac{\partial f_{\text{elem}}}{\partial z_j}\frac{\partial z_j}{\partial s} \tag{11.5.22}$$

This derivative can propagate through complicated functions and subroutines using the chain rule of differentiation. This propagation eventually produces the derivative of the structural response.

In the reverse mode, which corresponds to the adjoint method in the previous sections, the derivatives are computed backward through the computation. Due to the reverse procedure, this approach requires saving the entire function evaluation, which also requires a significant amount of memory.

Software that creates a computer program that calculates the derivatives of output of other computer programs is now available, and is applicable for small to medium-size programs.[24] The largest program that we found mentioned had about 140,000 lines.[30] Both first- and higher-order derivatives can be obtained. This approach was initially called automatic differentiation, but after a while it was realized that human intervention in the process is required in many cases to obtain a reasonably efficient code. So the name was generalized to computational differentiation.

There are several automatic differentiation tools widely available today, notably ADIFOR (Automatic Differentiation of Fortran[31] and ADOL-C for C/C++ programs.[32] In terms of implementation, there are two basic approaches to automatic differentiation — source code transformation and operator overloading. Source code transformation can be viewed as a precompiler that adds code for computing the derivatives. Operator overloading is available in modern computer languages, such as C++ and Fortran 90, that provide the ability to redefine the meaning of elementary operators (such as multiplication) for various classes of variables. By defining new variable types that have gradient objects associated with them, and overloading the elementary operators to produce gradients as well, we can transform the code without increasing its size substantially. ADOL-C and ADOL-F are examples of operator-overloading tools for automatic differentiation.

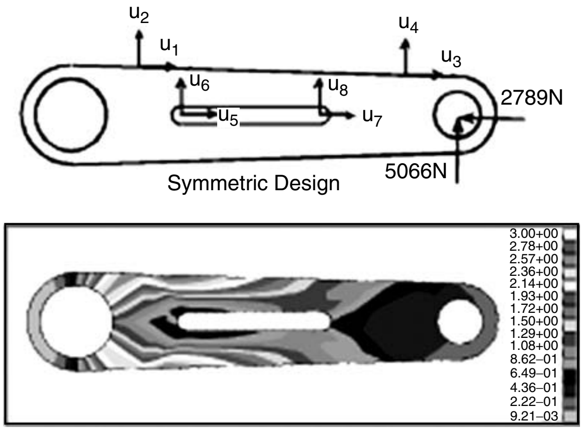


FIGURE 11.5.26 Design parameterization and mesh-free analysis results of a torque arm.

Examples

Torque Arm Model[33]

In order to show the accuracy of sensitivity calculation, a torque arm design problem, presented by Bennett and Botkin,[34] is used as a numerical example. The geometry of the torque arm (see Figure 11.5.26) is modeled using MSC/PATRAN,[35] and is represented by parametric coordinates. The displacement on the left hole is fixed, while the horizontal and vertical forces are applied at the center of the right hole. Design parameterization was performed by selecting the control points of the parametric curves such that the boundary curves move according to the design change. The design parameter was linked in order to obtain a symmetric design change. Design velocity vectors that represent the movement of particles in the direction of a given design parameter were computed by perturbing the parametric coordinates. This process is referred to as the isoparametric mapping method. Design parameterization and design velocity vector computation were also carried out by using a design sensitivity and optimization tool (DSO).[36] In all, eight design parameters were chosen in order to perturb the outer/inner boundary curves of the torque arm.

An automatic discretization procedure was used to discretize the torque arm structure. The domain of the torque arm was discretized by 478 degrees of freedom. The plane stress formulation is used with a thickness of 0.3 cm. The torque arm is made of steel with E = 207 GPa, and $\nu = 0.3$.

The structural analysis required 5.19 seconds, whereas by using one processor of HP Exemplar s-class workstation the sensitivity analysis required 4.55 ÷ 8 = 0.57 seconds per each design parameter. The efficiency of the sensitivity computation derives from the fact that sensitivity analysis uses the same constitutive matrix, already factorized from the response analysis stage.

The sensitivity coefficients of the performance measures, including the structural area and stresses, were computed based on the continuum approach. The highest stress values at 19 locations are selected as the performance measure, which will be served as constraints during optimization. Using a very small perturbation size, the accuracy of the sensitivity coefficients is compared with the finite difference results, as shown in Table 11.5.5. Very accurate sensitivity results are observed. In Table 11.5.5, the first column represents design parameters, the second column represents performance measures, that is, structural area and von Mises stress at eight integration zones. The third column $\Delta\psi$ denotes the first-order sensitivity results obtained from the forward finite difference method with a perturbation of $\tau = 10^{-6}$. The fourth column represents the sensitivity computation results from the method employed. As has been shown in the fifth column, the ratio between the third and the fourth columns are very close to 100%, which means the calculated sensitivity information is very accurate.

Figure 11.5.27 shows the design sensitivity plot of the von Mises stress performance with respect to design u_3. Such a sensitivity plot graphically illustrates the effect of design change to the performance

TABLE 11.5.5 Design Sensitivity Results and Comparison with Finite Difference Results

u	ψ	$\Delta\psi$	ψ'	$\Delta\psi/\psi' \times 100$
u_1	Area	.10361E-5	.10362E-5	100.00
	σ_{82}	-.62892E-7	-.62891E-7	100.00
	σ_{85}	-.17736E-8	-.17722E-8	100.08
	σ_{88}	-.88829E-7	-.88828E-7	100.00
	σ_{91}	-.11245E-6	-.11245E-6	100.00
	σ_{97}	-.77783E-7	-.77781E-7	100.00
	σ_{136}	-.15990E-6	-.15991E-6	100.00
	σ_{133}	.33665E-7	.33667E-7	100.00
	σ_{100}	-.67624E-7	-.67623E-7	100.00
u_3	Area	.10118E-5	.10119E-5	100.00
	σ_{82}	-.78084E-9	-.78177E-9	99.88
	σ_{85}	.14674E-9	.14678E-9	99.97
	σ_{88}	-.58752E-8	-.58748E-8	100.01
	σ_{91}	-.19387E-7	-.19387E-7	100.00
	σ_{97}	-.39358E-7	-.39357E-7	100.00
	σ_{136}	-.38821E-9	-.38886E-9	99.83
	σ_{133}	.41596E-9	.41525E-9	100.17
	σ_{100}	-.59788E-7	-.59787E-7	100.00
u_7	Area	-.20000E-5	-.20000E-5	100.00
	σ_{82}	.20682E-8	.20709E-8	99.87
	σ_{85}	.47302E-8	.47324E-8	99.95
	σ_{88}	.60386E-8	.60409E-8	99.96
	σ_{91}	.81475E-8	.81496E-8	99.98
	σ_{97}	.16225E-7	.16227E-7	99.99
	σ_{136}	-.78753E-9	-.78694E-9	100.08
	σ_{133}	.26136E-9	.26181E-9	99.83
	σ_{100}	.25006E-7	.25008E-7	99.99

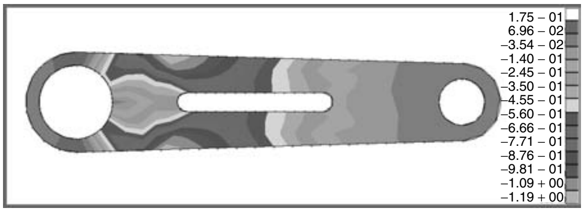


FIGURE 11.5.27 Von Mises stress sensitivity plot with respect to design u_3.

change, which provides very useful information in the interactive design process without invoking the automated design optimization procedure.

The Road Arm Model[33]

A road arm structure, as shown in Figure 11.5.28, transfers a force and torque from a road wheel to a suspension unit for a combat vehicle. The road arm model is discretized with 4365 degrees of freedom. The road arm is made of steel with $E = 206$ GPa, and $\nu = 0.3$. At the center of the right hole, a vertical force of 3736 N and a torque of 44,516 N-m are applied, whereas the displacement on the left hole is fixed. As was illustrated in Figure 11.5.28, the stress concentration appears in the left corner of the road arm. If the highest stress level in the left corner is considered as a reference value, then the dimension of the right corner cross section can be reduced, because this region has a large amount of safety margin.

Since two holes are connected to the road wheel and torsion bar, the dimension and the geometry of the holes are fixed. The design goal is to determine the dimension of the cross sections of the arm. The

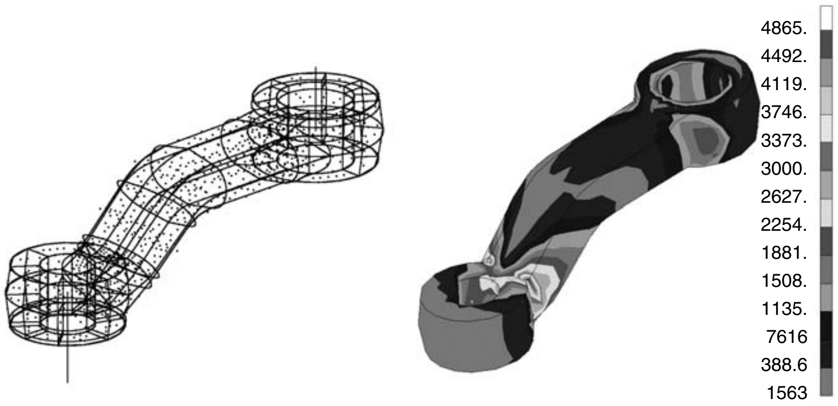


FIGURE 11.5.28 Discrete road arm model and simulation results (stress).

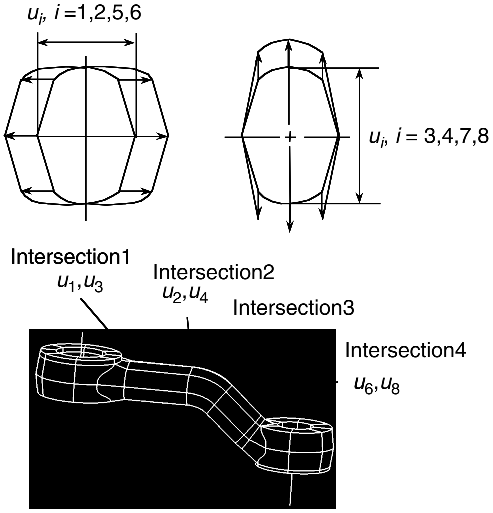


FIGURE 11.5.29 Design parameters for the rod arm model.

heights and widths of four sections are selected as design parameters (see Figure 11.5.29). Thus, a total of eight design parameters are considered in this example.

As the design variable changes, the boundary surface of the structure changes. At the same time, the discrete model also needs to be moved according to the design variable's change. Even if the design variable changes the boundary surface, it is recommended that the interior nodes be moved too. Otherwise, the accuracy of the perturbed model may deteriorate. The relation between a design variable and the motion of each node is denoted by the *design velocity field*. Figure 11.5.30 shows the design velocity field for two different design variables, u_2 and u_4, respectively. The arrows denote the magnitude and direction of nodal movement according to the corresponding design variable's change.

For a given design variable, the design sensitivity coefficients of various performance measures can be calculated using the design velocity field. Table 11.5.6 shows the design sensitivity coefficients, compared with the finite difference results. Displacement, stress, and volume of the structure are considered as performance measures. For example, zx4 represents the *x*-directional displacement at node 4, and sx10 is the *x*-directional normal stress at element 10. Very accurate sensitivity results are observed. This sensitivity information will be provided to the design optimization algorithm, to obtain the optimum design for given constraints.

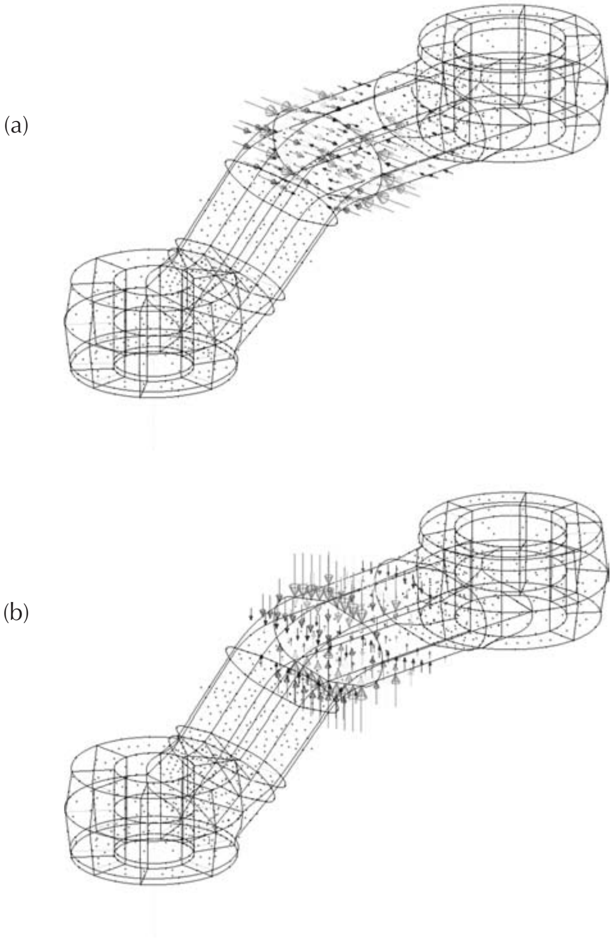


FIGURE 11.5.30 Design velocity field vectors: (a) design u_2; (b) design u_4.

TABLE 11.5.6 Design Sensitivity Results Compared with Finite Difference Method

Design	Perform.	ψ	$\Delta\psi$	$\psi'\Delta\tau$	$\Delta\psi/\psi'\Delta\tau \times 100$
u1	zx4	7.98472E–04	6.64019E–11	6.64028E–11	100.00
	zy8	–1.74039E–04	–1.50522E–11	–1.50526E–11	100.00
	sx10	1.38382E+01	–3.25110E–06	–3.26009E–06	99.72
	sy14	1.25479E+01	–1.40765E–05	–1.40762E–05	100.00
	sz18	–3.11292E+00	–1.79759E–06	–1.80037E–06	99.85
	Volume	4.68909E+02	–4.53781E–05	–4.53781E–05	100.00
u5	zx4	7.98472E–04	2.13594E–10	2.13594E–10	100.00
	zy8	–1.74039E–04	–7.76925E–11	–7.76924E–11	100.00
	sx10	1.38382E+01	4.83484E–07	4.83908E–07	99.91
	sy14	1.25479E+01	–4.81065E–06	–4.81087E–06	100.00
	sz18	–3.11292E+00	–4.00414E–06	–4.00603E–06	99.95
	Volume	4.68909E+02	–7.24906E–06	–7.24905E–06	100.00

Structural-Acoustic Design of a Vehicle[37]

As a last example, the design sensitivity analysis of a structural-acoustic problem is demonstrated. The goal is to estimate the sensitivity of a structure-borne noise caused by vibration of a vehicle. In order to calculate the noise level at the driver's ear position, sequential analysis procedure is employed. First, the

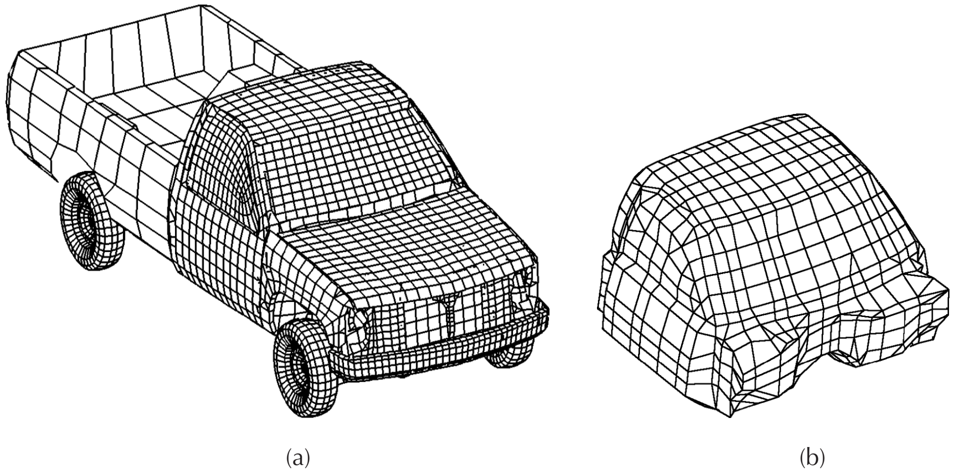


FIGURE 11.5.31 Computational model for structural-acoustic problem (a) finite element model of the vehicle; (b) boundary element model of the cabin compartment.

structural vibration problem is solved using the frequency response analysis. The vehicle structure is modeled using the finite element method, as shown in Figure 11.5.31(a). The sources of excitation are power-train vibration, wheel/terrain interaction, and a hydraulic pump load. Because of this additional source of excitation, vibration and noise is more significant than with a conventional power train. The frequency response analysis calculates the vibration velocity of vehicle's panels. Second, after solving for the panel velocity, the boundary element method is employed to calculate the pressure level in the cabin compartment using the panel velocity as a boundary condition. Figure 11.5.31(b) shows the boundary element model of the cabin compartment. In this example, the noise level of the passenger compartment is chosen as the performance measure, and vehicle-panel thicknesses are chosen as design variables. From the power-train analysis and rigid-body dynamic analysis, the harmonic excitations at 12 locations are obtained. The following frequency response analysis is carried out using MSC/NASTRAN to obtain the velocity response at eight frequencies, which correspond to the peak values of structure's velocity below 100 Hz:

$$[j\omega\mathbf{M} + \kappa\mathbf{K}]\{\mathbf{v}(\omega)\} = \{\mathbf{f}(\omega)\} \tag{11.5.23}$$

where $j = \sqrt{-1}$, ω is the excitation frequency, [**M**] is the structural mass matrix, $\kappa = (1 + j\phi)/j\omega$, ϕ is the structural damping coefficient, [**K**] is the stiffness matrix, {**v**} is the panel velocity vector, and {**f**} is the harmonic excitation force.

After solving the structure's velocity response, acoustic BEA is carried out using the cabin acoustic boundary element model, as shown in Figure 11.5.31(b). First, the emanating pressure from the panel is calculated using all panels' vibration velocity, as

$$[\mathbf{A}]\{\mathbf{p}_s\} = [\mathbf{B}]\{\mathbf{v}\} \tag{11.5.24}$$

where the matrices [**A**] and [**B**] are functions of geometry and $\{\mathbf{p}_s\}$ is the surface pressure of the panel. After calculating the surface pressure distribution, the sound pressure at the interior of the cabin can be calculated from

$$p = \{\mathbf{b}\}^T\{\mathbf{v}\} + \{\mathbf{e}\}^T\{\mathbf{p}_s\} \tag{11.5.25}$$

where the vectors {**b**} and {**e**} are functions of geometry.

TABLE 11.5.7 Acoustic Pressures at the Driver's Ear Position

Frequency (Hz)	Pressure (kg/mm·sec^2)	Phase Angle (Degree)
47.3	0.64275E–04	66.915
59.5	0.35889E–03	328.99
75.9	0.66052E–04	193.91
81.8	0.41081E–03	264.21
86.0	0.21629E–03	176.18
90.5	0.43862E–03	171.44
93.6	0.75627E–02	178.30
98.7	0.22676E–03	226.07

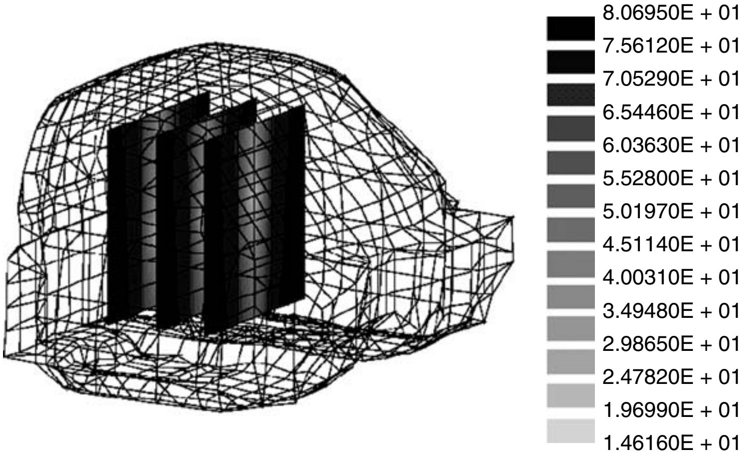


FIGURE 11.5.32 Acoustic pressure distribution inside the cabin compartment at 93.6 Hz.

Table 11.5.7 shows sound pressure levels at the driver's ear position. Since the sound pressure level at 93.6 Hz is significantly higher than at other frequencies, design modification is carried out at that frequency. In fact, this frequency corresponds to the fundamental frequency of the acoustic domain. Figure 11.5.32 shows the sound-pressure level inside the cabin compartment. The maximum sound pressure level at the driver's ear is 77.8 dB when the reference pressure of 2×10^{-8} kg/mm·s^2 is used.

In order to identify which panel thickness affects the sound pressure, the design sensitivity analysis has been performed. Forty design variables are selected in this example, which are grouped based on the manufacturing constraints. The adjoint method is employed in calculating the design sensitivity information because the number of performance measure (pressure at driver's ear position) is much smaller than the number of design variables. Table 11.5.8 shows the normalized sensitivity information for the 12 design variables whose effects are the most significant. The results show that a thickness change in the chassis component has the greatest potential for achieving a reduction in sound-pressure levels. Since the numerical integration process is carried out on each finite element, the element sensitivity information

TABLE 11.5.8 Normalized Sensitivity Values for Various Design Variables

Component	Sensitivity	Component	Sensitivity
Chassis	−1.0	Chassis MTG	−0.11
Left wheelhouse	−0.82	Chassis connectors	−0.10
Right door	0.73	Right fender	−0.07
Cabin	−0.35	Left door	−0.06
Right wheelhouse	−0.25	Bumper	−0.03
Bed	−0.19	Rear glass	0.03

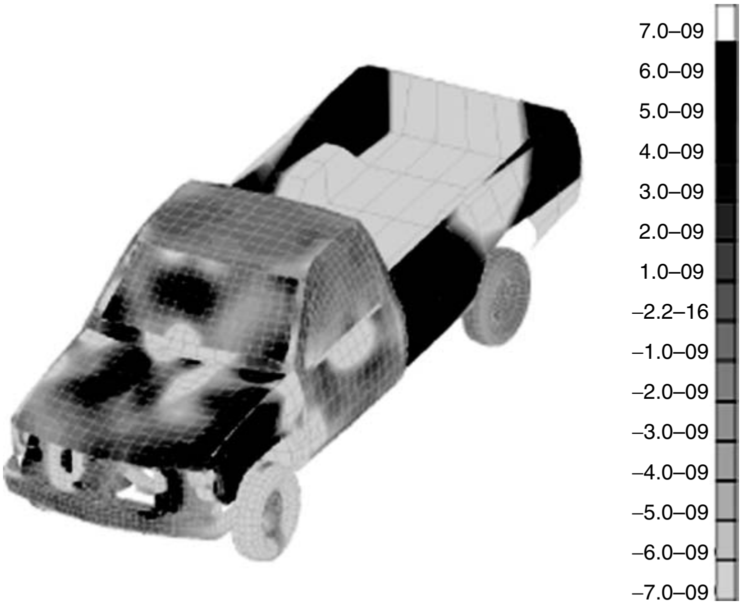


FIGURE 11.5.33 Design sensitivity contribution of each element to the sound pressure level.

can be calculated without any additional effort. Figure 11.5.33 plots the sensitivity contribution of the each element to the sound-pressure level. Such graphic-based sensitivity information is very helpful for the design engineer in determining the direction of the design modification.

As was shown in Table 11.5.8, the chassis component has the highest sensitivity for the sound-pressure level, which means that a change in the thickness of the chassis component is the most effective way to reduce the sound-pressure level. To see the effect of the chassis component, the thickness of the chassis is increased by 1.0 mm and the whole analysis process is repeated for the modified design. It turns out that the maximum value of the sound pressure is reduced from 77.8 dB to 75.0 dB.

Gradient-Based Design Optimization

When the design parameters are continuous, various numerical techniques have been developed to find the local optimum design systematically. Unfortunately, no mathematical theory exists that can find the global optimum design for general nonlinear functions. To find the optimum design, at least a possible candidate must exist within a feasible design region to satisfy problem constraints. Every design in the feasible region is an acceptable design, even if it is not the best one. The best design is usually the one that minimizes (or maximizes) the cost function of the design problem. Thus, the goal of the design optimization problem is to find a design that minimizes the cost function among all feasible designs. In this section, design optimization algorithms are briefly introduced. However, this brief discussion is by no means a complete treatment of optimization methods. For a more detailed treatment, see references 38, 39, and 40.

Most gradient-based optimization algorithms are based on the mathematical programming method, which requires the function values and sensitivity information at given design variables. Each algorithm has its own advantages and disadvantages. The performance of an optimization algorithm critically depends on the characteristics of the design problem and the types of cost and constraint functions.

The Linear Programming Method

The linear programming method can be used when cost and constraints are linear functions of the design variables.[41] Most structural design problems, however, are nonlinear with respect to their design variables. Thus, the linear programming method is not of much use for structural problems. However, a nonlinear problem can be solved by approximating a sequence of linear problems. The standard form of a linear programming problem is

$$\begin{aligned} &\text{minimize} \quad f = \mathbf{c}^T \mathbf{u} \\ &\text{subject to}\, \mathbf{A}\mathbf{u} = \mathbf{b} \\ &\qquad\qquad \mathbf{u} \geq \mathbf{0} \end{aligned} \tag{11.5.26}$$

where $\mathbf{c} = [c_1, c_2, \ldots, c_n]^T$ is the coefficient of the cost function, **A** is the $m \times n$ matrix, and **b** is the $m \times 1$ vector. Inequality constraints can be treated as equality constraints by introducing slack variables. Since all functions are linear, the feasible regions defined by linear equalities are convex, along with the cost function. Thus, if any optimum solution of Equation 11.5.26 exists, then it is a global minimum solution of the problem. The reason for introducing the linear problem here is that a very efficient method exists for solving linear programming problems, namely *the simplex method*. A positive feature of a linear programming problem is that the solution always lies on the boundary of the feasible region. Thus, the simplex method finds a solution by moving each corner point of the convex boundary.

Unconstrained Optimization Problems

When cost and/or constraints are nonlinear functions of the design, the design problem is called a *nonlinear programming method*, as contrasted to the linear programming method discussed in the previous section. Most engineering problems fall into the former category. Because the properties of nonlinear programming are nonlinear, this method is frequently solved using the numerical, rather than the analytical, method.

When there are no constraints on the design problem, it is referred to as an *unconstrained optimization problem*. Even if most engineering problems have constraints, these problems can be transformed into unconstrained ones by using the penalty method, or the Lagrange multiplier method. The unconstrained optimization problem sometimes contains the lower and upper limits of a design variable, since this type of constraint can be treated in simple way. The standard form of an unconstrained optimization problem can be written as

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{u}) \\ &\text{subject to} \;\; u_k^L \leq u_k \leq u_k^U, \qquad k = 1, \cdots, n \end{aligned}. \tag{11.5.27}$$

In the following subsections, numerical methods for solving Equation 11.5.27 are discussed.

The Steepest Descent Method

The numerical procedure for solving Equation 11.5.26 is an iterative update of design **u**. If $\mathbf{u}^k$ is the value of the design at the k-th iteration, then the new design at the $(k+1)$-th iteration can be obtained by

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \alpha \mathbf{d}^{k+1} \tag{11.5.28}$$

where $\mathbf{d}^{k+1}$ is called the descent direction and α is a step size, used to determine the amount of movement in the direction of $\mathbf{d}^{k+1}$. If the descent direction is given, then parameter α is determined by using the line search procedure to find the minimum value of a cost function in the descent direction. The steepest descent method uses the gradient of the cost function as the descent direction, such that

$$\mathbf{d}^{k+1} = -\frac{\partial f(\mathbf{u}^k)}{\partial \mathbf{u}^k} = -\nabla f(\mathbf{u}^k) \tag{11.5.29}$$

which is the design sensitivity of the cost function. This method suffers from a slow convergence near the optimum design, since it does not use any information from the previous design, and only first-order information of the function is used. Note that $\mathbf{d}^k$ and $\mathbf{d}^{k+1}$ are always orthogonal, such that a zigzagging pattern appears in the optimization process.

The Conjugate Gradient Method

The conjugate gradient method developed by Fletcher and Reeves[42] improves the rate of slow convergence in the steepest descent method by using gradient information from the previous iteration. The difference in this method is the computation of $\mathbf{d}^{k+1}$ in Equation 11.5.28. The new descent direction is computed by

$$\mathbf{d}^{k+1} = -\nabla f(\mathbf{u}^k) + \beta_k^2 \mathbf{d}^k \tag{11.5.30}$$

where

$$\beta_k = \frac{\left\|\nabla f(\mathbf{u}^k)\right\|}{\left\|\nabla f(\mathbf{u}^{k-1})\right\|} \tag{11.5.31}$$

and where the first iteration is the same as Equation 11.5.28. This method tends to select the descent direction as a diagonal of two orthogonal steepest descent directions, such that a zigzagging pattern can be eliminated. This method always has better convergence than the steepest descent method.

The Newton Method

The previous methods we have examined use first-order information (first-order design sensitivity) of the cost function to find the optimum design, which is called linear approximation. The Newton method uses second-order information (second-order design sensitivity) to approximate the cost function as a quadratic function of the design. The major concern is how to compute the second-order design sensitivity (or Hessian) matrix. Let us define the Hessian matrix as second-order design sensitivity, as

$$\mathbf{H}(\mathbf{u}^k) \equiv \left[\frac{\partial f(\mathbf{u}^k)}{\partial u_i^k \partial u_j^k}\right], \quad i, j = 1, \cdots, n \tag{11.5.32}$$

The new design can then be determined, as

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\mathbf{u}^{k+1} \tag{11.5.33}$$

where

$$\Delta\mathbf{u}^{k+1} = -\mathbf{H}(\mathbf{u}^k)^{-1}\nabla f(\mathbf{u}^k) \tag{11.5.34}$$

If the current estimated design $\mathbf{u}^k$ is sufficiently close to the optimum design, then Newton's method will show a quadratic convergence. However, the greater the number of design variables, the greater the cost of computing $\mathbf{H}(\mathbf{u}^k)$ in Equation 11.5.32. In addition, Newton's method does not guarantee a convergence. Thus, several modifications are available. For example, the design update algorithm in Equation 11.5.33 can be modified to include a step size by using a line search, as in Equation 11.5.28.

The Quasi-Newton Method

Although Newton's method has a quadratic convergence, the cost of computing the Hessian matrix, and the lack of a guaranteed convergence, are drawbacks to this method. The quasi-Newton method has an advantage over the steepest descent method and Newton's method: it only requires first-order sensitivity information, and it approximates the Hessian matrix to speed up the convergence.

The DFP (Davidon-Fletcher-Powell)[42] method approximates the inverse of the Hessian matrix using first-order sensitivity information. By initially assuming that the inverse of the Hessian is the identity matrix, this method updates the inverse of the Hessian matrix during design iteration. A nice feature of this method is that the positive definiteness of the Hessian matrix is preserved.

The BFGS (Broydon-Fletcher-Goldfarb-Shanno)[43] method updates the Hessian matrix directly, rather than updating its inverse as with the DFP method. Starting from the identity matrix, the Hessian matrix remains positive definite if exact line search is used.

Constrained Optimization Problems

Most engineering problems have constraints that must be satisfied during the design optimization process. These two types of constraints are handled separately: equality and inequality constraints. The standard form of the design optimization problem in constrained optimization can be written as

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{u}) \\ &\text{subject to} h_i(\mathbf{u}) = 0, \quad i = 1,\cdots,p \\ &\qquad g_j(\mathbf{u}) \le 0, \quad j = 1,\cdots,m \\ &\qquad u_l^L \le u_l \le u_l^U, l = 1,\cdots,n \end{aligned} \tag{11.5.35}$$

The computational method to find a solution to Equation 11.5.35 has two phases: first, to find a direction **d** that can reduce the cost $f(\mathbf{u})$ while correcting for any constraint violations that are violated; and second, to determine the step size of movement α in the direction of **d**.

Sequential Linear Programming (SLP)

The SLP method approximates the nonlinear problem as a sequence of linear programming problems such that the simplex method described earlier may be used to find the solution to each iteration. By using function values and sensitivity information, the nonlinear problem in Equation 11.5.35 is linearized in a similar way as Taylor's expansion method in the first order, as

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{u}^k) + \nabla f^T \Delta\mathbf{u}^k \\ &\text{subject to} \quad h_i(\mathbf{u}^k) + \nabla h_i^T \Delta\mathbf{u}^k = 0, \quad i = 1,\cdots,p \\ &\qquad g_i(\mathbf{u}^k) + \nabla g_i^T \Delta\mathbf{u}^k \le 0, j = 1,\cdots,m \\ &\qquad u_l^L \le u_l \le u_l^U, \qquad l = 1,\cdots,n \end{aligned} \tag{11.5.36}$$

Since all functions and their sensitivities at $\mathbf{u}^k$ are known, the linear programming problem in Equation 11.5.36 can be solved using the simplex method for $\Delta\mathbf{u}^k$. Even if the sensitivity information is not used to solve a linear programming problem, design sensitivity information is required in order to approximate the nonlinear problem as a linear one with SLP. In solving Equation 11.5.36 for $\Delta\mathbf{u}^k$, the move limit $\Delta\mathbf{u}_L^k \le \Delta\mathbf{u}^k \le \Delta\mathbf{u}_U^k$ is critically important for convergence.

Sequential Quadratic Programming (SQP)

Compared to previous methods, which use first-order sensitivity information to determine the search direction **d**, SQP solves a quadratic subproblem to find that search direction, which has both quadratic cost and linear constraints:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{u}^k) + \nabla f^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{H}\mathbf{d} \\ &\text{subject to} \quad h_i(\mathbf{u}^k) + \nabla h_i^T \mathbf{d} = 0, \quad i = 1,\cdots,p \\ &\qquad g_j(\mathbf{u}^k) + \nabla g_j^T \mathbf{d} \le 0, j = 1,\cdots,m \end{aligned} \tag{11.5.37}$$

This special form of the quadratic problem can be effectively solved, for example, by using the Kuhn-Tucker condition and the simplex method. Starting from the identity matrix, the Hessian matrix **H** is updated at each iteration by using the aforementioned methods in unconstrained optimization algorithms. The advantage of solving Equation 11.5.37 in this way is that for positive definite **H** the problem is convex and the solution is unique. Moreover, this method does not require the move limit as in SLP.

The Constrained Steepest Descent Method

In the unconstrained optimization process described earlier, the descent direction **d** is obtained from the cost function sensitivity. When constraints exist, this descent direction has to be modified in order to include their effects. If constraints are violated, then these constraints are added to the cost function using a penalty method. Design sensitivity of the penalized cost function combines the effects of the original cost function and the violated constraint functions.

The Constrained Quasi-Newton Method

If the linear approximation of constraints in SQP is substituted for a quadratic approximation, then the convergence rate of Equation 11.5.37 will be improved. However, solving the optimization problem for quadratic cost and constraints is not an easy process. The constrained quasi-Newton method combines the Hessian information of constraints with the cost function by using the Lagrange multiplier method. Nevertheless, it is still necessary to compute the constraint function Hessian. The main purpose of the constrained quasi-Newton method is to approximate the Hessian matrix by using first-order sensitivity information. The extended cost function is

$$L(\mathbf{u},\mathbf{v},\mathbf{w}) = f(\mathbf{u}) + \sum_{i=1}^{p} v_i h_i(\mathbf{u}) + \sum_{j=1}^{m} w_i g_i(\mathbf{u}) \tag{11.5.38}$$

where both $\mathbf{v} = [v_1, v_2, \ldots, v_p]^T$ and $\mathbf{w} = [w_1, w_2, \ldots, w_m]^T$ are the Lagrange multipliers for equality and inequality constraints, respectively. Note that $\mathbf{w} > \mathbf{0}$. Let the second-order design sensitivity of L be $\nabla^2 L$. The extended quadratic programming problem of Equation 11.5.37 thus becomes

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{u}^k) + \nabla f^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \nabla^2 L \mathbf{d} \\ &\text{subject to} \quad h_i(\mathbf{u}^k) + \nabla h_i^T \mathbf{d} = 0, \quad i = 1, \cdots, p \\ &\qquad\qquad g_j(\mathbf{u}^k) + \nabla g_j^T \mathbf{d} \le 0, \quad j = 1, \cdots, m \\ &\qquad\qquad w_l \ge 0, \qquad\qquad l = 1, \cdots, m \end{aligned} \tag{11.5.39}$$

The Feasible Direction Method

The feasible direction method is designed to allow design movement within the feasible region in each iteration. Based on the previous design, the updated design reduces the cost function and remains in the feasible region. Since all designs are feasible, a design at any iteration can be used, even if it is not an optimum design. Because this method uses the linearization of functions as in SLP, it is difficult to maintain nonlinear equality constraints. Thus, this approach is used exclusively for inequality constraints. Search direction **d** can be found by solving the following linear subproblem:

$$\begin{aligned} &\text{minimize} \quad \beta \\ &\text{subject to} \quad \nabla f^T \mathbf{d} \le \beta \\ &\qquad\qquad \nabla g_i^T \mathbf{d} \le \beta, \quad i = 1, \cdots, m_{active} \\ &\qquad\qquad -1 \le d_j \le 1, \quad j = 1, \cdots, n \end{aligned} \tag{11.5.40}$$

where m_{active} is the number of active inequality constraints. After finding a direction **d** that can reduce cost function and maintain feasibility, a line search is used to determine step size α.

The Gradient Projection Method

The feasible direction method solves the linear programming problem to find the direction of the design change. The gradient projection method, however, uses a simpler method for computing this direction. The direction obtained by the steepest descent method is projected on the constraint boundary, such that the new design can move along the constraint boundary. Thus, the direction of the design change reduces the cost function while maintaining the constraint along its boundary. For a general nonlinear constraint, however, a small movement along the tangent line of the boundary will violate this constraint. Thus, in actual implementation, a correction algorithm has to be followed. The gradient projection method behaves well when the constraint boundary is moderately nonlinear.

Examples

The Torque Arm Model

The torque arm model is used here to demonstrate the design optimization. The design optimization problem is formulated in such a way that the total area of the structure is minimized with respect to its shape design parameters, with design constraints defined as the second invariant of the stress tensors (von Mises stress), as

$$\begin{array}{ll} \text{minimize} & mass \\ \text{subject to} & \sigma_{MAX} \le 800\, MPa \end{array} \tag{11.5.41}$$

As has been shown in Figure 11.5.26, the maximum stress at the initial design was 305 MPa. Thus, the design that meets the constrained boundary in Equation 11.5.41 is far away from the initial design, which means the amount of shape change during design optimization is significantly large.

For design optimization, the sequential quadratic programming method described earlier has been used employing a commercially available optimization program.[39] The structural analysis provides the function values (i.e., mass and stress), whereas the sensitivity analysis provides the gradient information to the optimization algorithm. The design optimization problem converges after 20 numbers of iteration. Figure 11.5.34 shows the structural analysis results at optimum design where the stress constraints along the upper side of torque arm became active. The left diameter of the interior slot significantly increases to reduce the structural mass, whereas the right diameter is slightly decreased. The vertical position of the right boundary (u_4) is significantly reduced so that the thickness of the torque arm becomes constant.

Table 11.5.9 shows the values of design variables at the initial and optimum designs. The lower and upper bounds are selected such that the structure maintains its topology during design optimization. All design variables start from zero, which means the values of design variables are relative change of its coordinates. Four design variables are changed up to their lower or upper bounds.

Figure 11.5.35 provides an optimization history of the mass function. Through optimization, the structural mass was reduced from 0.878 kg to 0.421 kg (47.9%). The highest stress value initially, 305 MPa around the left hole, shifts to 800 MPa around the upper frame at optimum design. A total of 41 response analyses and 20 sensitivity analyses were carried out during 20 optimization iterations. When

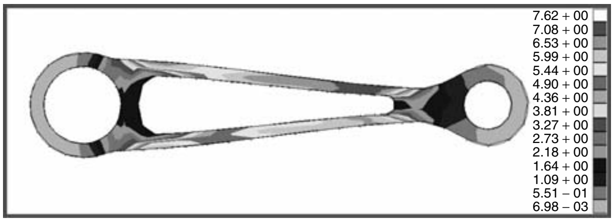


FIGURE 11.5.34 Analysis result at optimum design.

TABLE 11.5.9 Design Variables at the Initial and Optimum Designs

Design	Lower Bound	Initial Design	Optimum Design	Upper Bound
u_1	−3.000	0.0	−3.000	2.000
u_2	−0.500	0.0	−0.500	2.000
u_3	−2.000	0.0	−0.589	2.000
u_4	−3.000	0.0	−2.700	2.000
u_5	−4.500	0.0	−4.490	2.000
u_6	−0.500	0.0	2.000	2.000
u_7	−2.000	0.0	4.460	5.000
u_8	−0.500	0.0	−0.00714	2.000

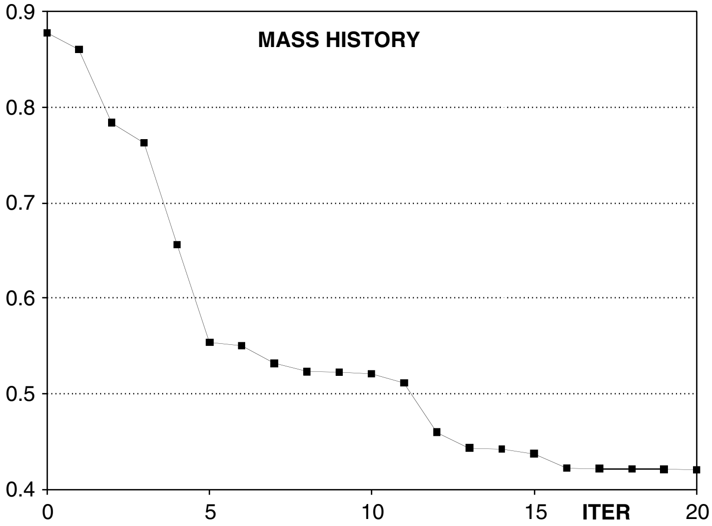


FIGURE 11.5.35 Histories of design optimization.

the finite difference methods are used with a re-meshing process,[34] the optimization process converged at 45 iterations with eight re-meshing processes. Thus, this approach reduces the cost of design more than 50%, without even mentioning the cost related to the re-meshing process.

The Road Arm Model

As a second example, the design optimization is carried out to minimize the structural weight of the road arm, while maintaining the maximum stress level. The same algorithm as with the torque arm model is used. Design optimization problem converges after eight iterations. Figure 11.5.36 compares the structural analysis result at the initial and optimum designs. The structural weight at the optimum design is reduced by 23% compared to the initial weight. Since the stress concentration appears in the left corner at the initial design, the optimization algorithm tried to reduce the cross section of the right corner so that both parts may have the same level of stress values. Figure 11.5.37 shows the optimization history of cost function and design variables. Due to the accurate sensitivity information, the optimization algorithm converges rather quickly. Also, the small design change can be another explanation of the fast convergence.

Gradient-Free Design Optimization

Genetic Algorithms

Genetic algorithms (GA) are inspired by Darwin's principle of evolution, which states that a population of individuals is capable of adapting to its environment because individuals who possess traits that make them less vulnerable than others are more likely to have descendents and therefore to pass on their desirable

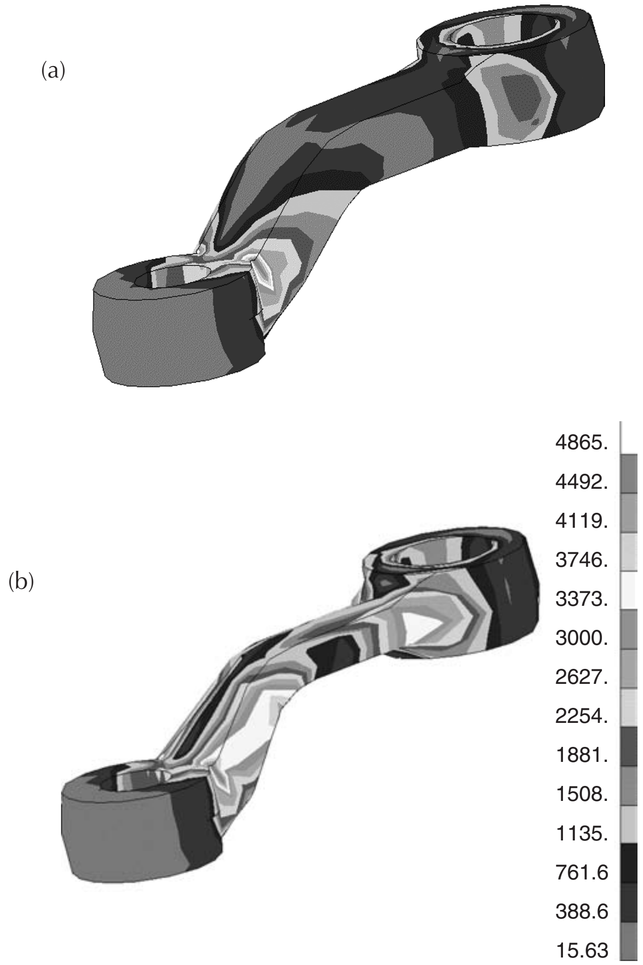


FIGURE 11.5.36 Structural analysis results (a) initial design; (b) optimum design.

traits to the next generation. One can think of this process of adaptation as an optimization process that probabilistically creates fitter individuals through selection and recombination of good characters. Genetic algorithms are simplified computer models of evolution, where the environment is emulated by the objective function to maximize, and the structure to optimize plays the role of the individuals.

A flowchart of a genetic algorithm is presented in Figure 11.5.38. GAs start by initializing a population of individuals at random. Each individual encode of a particular candidate structure in the form of one or several chromosomes, which are strings of finite length. Then the objective function of each individual, called the fitness function in the context of evolutionary computation, is computed. The fitness of each individual determines its probability of being selected for reproduction. Instead of moving from one design point to another, the search is based on a population of design points that evolve from one generation to another. Recombination and mutation operators are then applied to the selected individuals (the parents) to create a population of children. Finally a survival rule determines the individuals among the parent and child population that will be kept to form the new population.

When to Use Genetic Algorithms?

Genetic algorithms are computationally expensive compared to deterministic algorithms. Consequently, their domain of application consists of problems that would be difficult to solve by gradient-based algorithms, or problems that would be out of range for these algorithms. Some of the main advantages of GAs are the following:

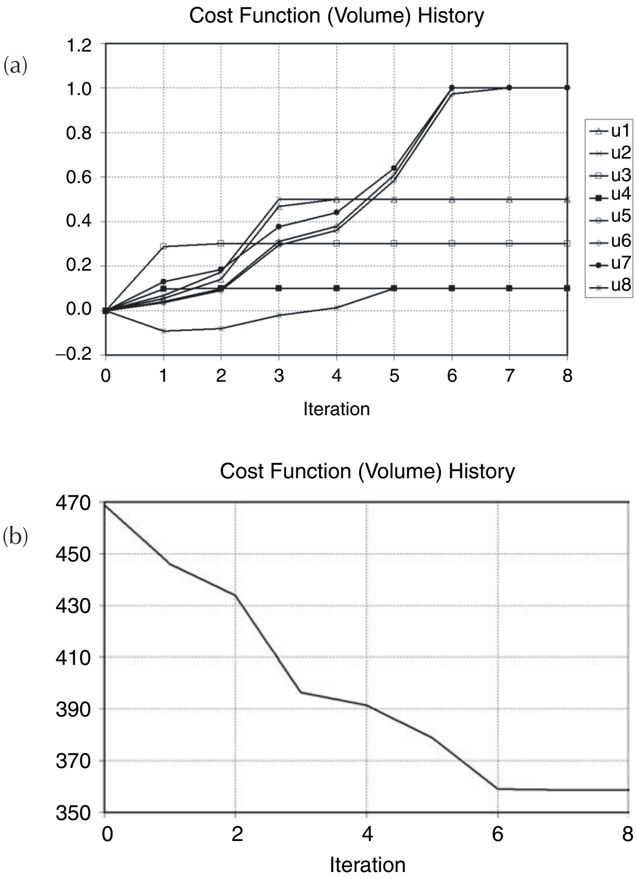


FIGURE 11.5.37 Design optimization history (a) cost function history; (b) design variable history.

- GAs do not require continuity or differentiability of the objective function. They do not need gradients.
- They are robust: they are very insensitive to noise.
- They are modular and therefore portable; because the evolutionary mechanism is separate from the problem representation, they can be transferred from problem to problem.
- They are particularly efficient on discrete and combinatorial problems, which are typically difficult to solve by conventional algorithms; they can be used for problems involving both discrete and continuous variables.
- Because they explore the design space with populations, GAs are “naturally” amenable to parallelization.

Examples of Applications

GAs have been used to solve a wide variety of engineering problems that would be difficult to tackle with other methods, such as scheduling problems, distribution network optimization, plant or product design, and real-time control of industrial processes and communications networks. A practical example of application is the optimization of the lay-up of composite laminate, where the goal is to determine the optimal fiber orientation of each of the layers of the laminate. This problem is difficult to solve by gradient-based algorithms because the objective function is often multimodal, and the fiber orientations have to be chosen from a set of discrete values to meet manufacturability requirements. On these problems, GAs prove to be very effective design tools.

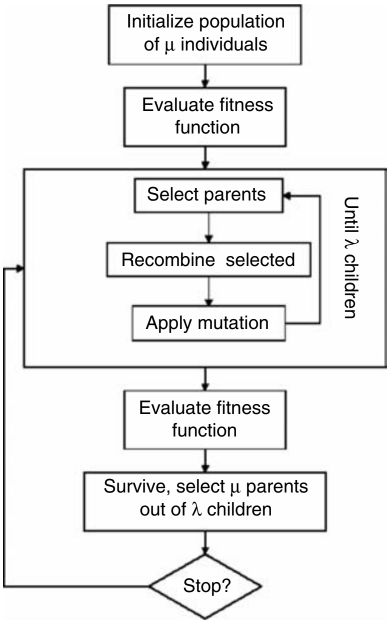

Initialize population of μ individuals
Evaluate fitness function
Select parents
Recombine selected
Apply mutation
Until λ children
Evaluate fitness function
Survive, select μ parents out of λ children
Stop?

FIGURE 11.5.38 Genetic algorithm.

Simulated Annealing

Simulated annealing (SA) is based on a simple algorithm introduced by Metropolis et al.[44] to efficiently predict the behavior of a collection of atoms in equilibrium at a given temperature. The Metropolis approach simulates the cooling process by means of an iterative algorithm, in which each atom is given a small random displacement and the resulting change of energy ΔE is computed. If $\Delta E \leq 0$ the displacement is accepted and the resulting configuration is used as the initial point for the next iteration. If $\Delta E > 0$ a probability of acceptance is calculated with the Boltzman factor:

$$P\left(\Delta E\right) = e^{-\left(\frac{\Delta E}{k_b T}\right)}$$

If ΔE is less than $P(\Delta E)$ the new configuration is accepted and used in the next iteration. This criterion ensures that the system will eventually evolve into a Boltzman equilibrium distribution.

By using the Metropolis procedure with a cost function in place of system energy, the simulated annealing algorithm is obtained.[45] The configuration of atoms will then be a combination of design parameters, which are evaluated in the same manner.

The temperature T becomes a control parameter with the same units as the cost function, and is used to first “melt” the system at a high temperature, with a population of possible parameter configurations being generated. The iterative simulation is then run at this temperature until the system is considered to be in equilibrium, and only then is the temperature reduced by a small fraction. This cooling and equilibrium process continues until no further improvement becomes possible and the system is considered “frozen” or crystallized, with the design parameters at this condition optimal. The temperature reduction sequence and number of iterations allowed for the system to reach equilibrium are considered analogues to an annealing schedule.

An important difference from iterative improvement procedures such as gradient-based methods is that the Metropolis method need not get stuck since transitions out of a local optimum are always possible at a nonzero temperature.

Applications of this algorithm include the well-known traveling salesman problem,[46] and routing and layout of electrical connections on computers,[44] among others. Like genetic algorithms, this population-based method is relatively insensitive to noise, and doesn't require gradient information. A major drawback to this method, however, is the problem dependency of algorithm parameters, for example, it is important to use an annealing schedule that is tailored to the type of optimization problem being solved; otherwise, the optimizer will perform poorly. Various statistical methods have been proposed, all with limited success, to obtain optimum cooling rates to avoid entrapment in local optima.

References

1. Bendsoe, M.P.1995. *Optimization of Structural Topology, Shape, and Materials*. Springer-Verlag, Berlin.
2. Clark and Fujimoto 1991. *Product Development Performance*. Harvard Business School Press, Boston.

AU: Need 1st initials

3. Zienkiewicz, O.C. 1977. *The Finite Element Method*. McGraw-Hill, New York.
4. Haug, E.J., Choi, K.K., and Komkov, V. 1986. *Design Sensitivity Analysis of Structural Systems*. Academic Press, London.
5. Haftka, R.T. and Grandhi, R.V., Structural shape optimization — a survey. *Computer Methods in Appl. Mech. and Eng.*, 57, 91–106, 1986.
6. Ding, Y., Shape optimization of structures — a literature survey. *Comput. and Structures*, 24, 6, 985–1004, 1986.
7. Bhavikatti, S.S. and Ramakrishnan, C.V., Optimum shape design of rotating disks. *Comput. and Structures*, 11, 397–401, 1980.
8. Prasad, B. and Emerson, J.F., Optimal structural remodeling of multi-objective systems. *Comput. and Structures*, 18, 4, 619–28, 1984.
9. Kristensen, E.S. and Madsen, N.F., On the optimum shape of fillets in plates subjected to multiple in-plane loading cases. *Int. J. for Numerical Methods in Eng.*, 10, 1007–19, 1976.
10. Pedersen, P. and Laursen, C.L., Design for minimum stress concentration by finite elements and linear programming. *J. of Structural Mech.*, 10, 375–91, 1982.
11. Yang, R.J. and Choi, K.K., Accuracy of finite element based shape design sensitivity analysis. *J. of Structural Mech.*, 13, 223–39, 1985.
12. Luchi, M.L., Poggialini, A., and Persiani, F., An interactive optimization procedure applied to the design of gas turbine discs. *Comput. and Structures*, 11, 629–37, 1980.
13. Weck, M. and Steinke, P., An efficient technique in shape optimization. *J. of Structural Mech.*, 11, 433–49. 1983–84.
14. Braibant, V., Fleury, C., and Beckers, P. 1983. *Shape Optimal Design: An Approach Matching CAD and Optimization Concepts*, Aerospace Laboratory of the University of Liege, Belgium.
15. Braibant, V. and Fleury, C., *Shape optimal design using B-spline. Computer Methods in Appli. Mech. and Eng.*, 44, 247–67, 1984.
16. Yao, T.M. and Choi, K.K., 3-D shape optimal design and automatic finite element regridding. Int. *J. for Numerical Methods in Eng.*, 28, 369–84, 1989.
17. Yao, T.M. and Choi, K.K., Shape optimal design of an arch dam. *ASME J. of Structural Eng.*, 115, 9, 2401–5, 1989.
18. Choi, K.K. and Yao, T.M., 3-D shape modeling and automatic regridding in shape design sensitivity analysis. *Sensitivity Analysis in Engineering*, NASA Conference Publication 2457, pp. 329–45, 1987.
19. Akgün, M.A., Garcelon, J.H., and Haftka, R.T., Fast exact linear and nonlinear structural reanalysis and the Sherman-Morrison-Woodbury formulas. *Int. J. Numerical Methods in Eng.*, 50, 7, 1587–1606, 2001.
20. Yoon, B.G. and Belegundu, A.D., Iterative methods for design sensitivity analysis. *AIAA J.*, 26, 11, 1413–15, 1988.

21. Kirsch, U. and Lui, S., Structural reanalysis for general layout modifications. *AIAA J.,* 35, 382–88, 1997.
22. Oral, S., An improved semianalytical method for sensitivity analysis. *Structural Optimization*, 11, 67–69, 1996.
23. Hörnlein, H.R.E.M., Effiziente semi-analytische gradientenberechnung in der strukturoptimierung. Z. Angew. *Math. Mech*., 81, s669–s670, 2000.
24. Barthelemy, B.M. and Haftka, R.T., Accuracy Analysis of the Semi-analytical Method for Shape Sensitivity Calculation. 29th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics and Materials Conference, 1988.
25. Olhoff, N., Rasmussen, J., and Lund, E., A method of exact numerical differentiation for error elimination in finite-element-based semianalytical shape sensitivity analyses. *Mech. of Structures and Machines*, 21, 1, 1–66, 1993.
26. Arora, J.S., An exposition of the material derivative approach for structural shape sensitivity analysis. *Computer Methods in Appl. Mech. and Eng.*, 105 ,41–62, 1993.
27. Phelan, D.G. and Haber, R.B., Sensitivity analysis of linear elastic systems using domain parameterization and a mixed mutual energy principle. *Computer Methods in Appl. Mech. and Eng.*, 77, 31–59, 1989.
28. Yang, R.J. and Botkin, M.E., Accuracy of the domain method for the material derivative approach to shape design sensitivities, in *Sensitivity Analysis in Engineering*, 1987, NASA Conference Publication 2457, pp. 347–53.
29. Ozaki, I. and Terano, T., Applying an automatic differentiation technique to sensitivity analysis in design optimization problems. *Finite Elements in Analysis and Design*, 14, 143–151, 1993.
30. Walsh, J.L., Young, K.C., Tarzanin, F.J., Hirsh, J.E., and Young, D.E., Optimization issues with complex rotorcraft comprehensive analysis. 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization (September 2–4, 1998), Volume Part 2, AIAA 98-4889.
31. Bischof, C., Carle, A., Khademi, P., and Mauer, A., The adifor 2.0 system for the automatic differentiation of fortran 77 programs. *IEEE Computational Science & Eng*., 3, 18–32, 1996.
32. Griewank, A., Juedes, D., and Utke, J., Adol-c, a package for the automatic differentiation of algorithms written in C/C++. *TOMS*, 22, 2, 131–67, 1996.
33. Kim, N.H., Choi, K.K., and Botkin, M.E., Numerical method of shape optimization using meshfree method. *Structural and Multidisciplinary Optimization*, 24, 6, 418–29, 2003.
34. Bennett, J. A. and Botkin, M.E., Structural shape optimization adaptive mesh refinement. *AIAA J.*, 23, 458–64, 1985.
35. *MSC/PATRAN User's Guide*. The MacNeal–Schwendler Corp., Los Angeles, CA, 1999.
36. Chang, K.H., Choi, K.K., Tsai, C.S., Chen, C.J., Choi, B.S., and Yu, X.,. Design sensitivity analysis and optimization tool (DSO) for shape design applications. *Computing Syst. in Eng*. 6, 151–175, 1995.
37. Kim, N.H., Dong, J., Choi, K.K., Vlahopoulos, N., Ma, Z.D., Castanier, M.P., and Pierre, C., Design sensitivity analysis for a sequential structural-acoustic problem. *J. Sound and Vibration*, 263, 3, 569–91, 2003.
38. Arora, J.S. 1999. *Introduction to Optimum Design*. McGraw-Hill, New York.
39. Vanderplaats, G.N. 1999. *Numerical Optimization Techniques for Engineering Design with Applications*. Vanderplaats Research & Development Inc., Colorado Springs, CO.
40. Haftka, R.T. and Karmat, M.P. 1985. *Elements of Structural Optimization*. Nijhoff Publishers, Netherlands.
41. Luenberger, D.G. 1984. *Linear and Nonlinear Programming*. Addison-Wesley, Boston.
42. Fletcher, R. and Reeves, R.M., Function minimization by conjugate gradients. *Computer J.*, 7, 149–60, 1964.
43. Fletcher, R. and Powell, M.J.D., A rapidly convergent descent method for minimization. *Computer J.*, 6, 163–80, 1963.

44. Metropolis,N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21, 6, 1087–92, 1953.
45. Kirkpatrick, S., Gelatt C.D. Jr., and Vecchi, M.P., Optimization by simulated annealing. In *Science*, 220, 4598, 671–80, 1983.
46. Cerny, V., Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, *J. Opt. Theory Appl.*, 45, 1, 41–51, 1985.