*Structural Safety*, 7 (1990) 131–148
Elsevier



# COMPUTATIONAL DESIGN OPTIMIZATION: A REVIEW AND FUTURE DIRECTIONS *

**Jasbir S. Arora**

*Optimal Design Laboratory, College of Engineering, The University of Iowa, Iowa City, IA 52242-1593 (U.S.A.)*

(Accepted May 1989)

**Key words**: optimization methods; nonlinear problems; review; computational aspects; engineering design.

## ABSTRACT

*A mathematical model for design optimization of engineering systems is defined. Computational algorithms to treat the model are reviewed and their features are discussed. The attributes of a good algorithm are given. Sequential quadratic programming algorithms that generate and use the approximate Hessian of the Lagrange function to calculate the search direction are the most recent methods. They are the most reliable methods among the available ones. Several other computational aspects, such as robust implementation of algorithms, use of a knowledge base, interactive use of optimization, and use of a database and database management system, are discussed. Recent developments in the field and future directions are presented.*

## 1. INTRODUCTION

The term *computational design optimization* (CDO) is introduced in this paper to encompass all the numerical methods used for the optimum design of engineering systems. The field has matured during the 1970s and 1980s and is beginning to be utilized in realistic design applications. Numerous methods have been developed and evaluated in the literature. Many papers have been published that show applications of the methods. It is not possible to include in this paper a detailed review of all the methods and their variations, or reference to all the published articles. Only a few algorithms that have been applied successfully to engineering design problems are included. The following list of review articles, books and proceedings and the literature cited in them contain work of almost all the researchers in the field:

* Presented at the Workshop on *Research Needs for Applications of System Reliability Concepts and Techniques in Structural Analysis, Design and Optimization*, Boulder, CO, September 12–14, 1988.

0167-4730/90/$03.50 

*Review articles*: Venkayya [1], Schmit [2], Ashley [3], Vanderplaats [4], Olhoff and Taylor [5], Arora and Belegundu [6], Belegundu and Arora [7], Arora and Thanedar [8], Haftka and Grandhi [9], Adelman and Haftka [10], Levy and Lev [11].

*Books*: Himmelblau [12], Spillers [13], Wilde [14], Haug and Arora [15], Gill, Murray and Wright [16], Kirsh [17], Iyengar and Gupta [18], Reklaitis, Ravindran and Ragsdell [19], Rao [20], Vanderplaats [21], Osyczka [22], Haftka and Kamat [23], Haug, Choi and Komkov [24], Shoup and Mistree [25], Papalambros and Wilde [26], Gajewski and Zyczkowski [27], Rozvany [28], Haslinger and Neittaanmäki [29], Arora [30].

*Proceedings*: Gallagher and Zienkiewicz [31], Haug and Cea [32], Lev [33], Morris [34], Eschenauer and Olhoff [35], Atrek, Gallagher, Ragsdell and Zienkiewicz [36], Sobieski [37], Ragsdell [38], Schittkowski [39], Gero [40], Bennett and Botkin [41], Mota Soares [42], Adelman and Haftka [43], Rozvany and Karihaloo [44], Brandt [45].

Engineering design problems are usually quite complex. In addition, as more powerful computers become available, the desire to solve more complex and larger problems also grows. This can lead to numerical difficulties if stable, reliable and efficient optimization algorithms are not used. Note that all the algorithms discussed in this paper converge to a local optimum point only. A global optimum is possible for convex problems or through exhaustive search of the design space.

The purpose of this paper is to present the basic concepts of computational methods for design optimization and review them in the context of application to large-scale complex problems involving massive calculations. The paper is an updated report on the state-of-the-art presented previously by Arora and Thanedar [8]. More recent literature is reviewed and some new algorithms are included. Some of the material from Ref. [8] is repeated and updated here for completeness. Based on these reviews, some directions for further research in the field are given.

## 2. DESIGN OPTIMIZATION MODEL

The formulation of the problem is an important step in the optimum design of a system. Problems of static and dynamic response, shape optimization of mechanical and structural systems, reliability-based design, optimal control of systems and many others can be formulated as *nonlinear programming* (NLP) problems [8,15,30,46–49]. These problems usually require very large analysis models for accurate response prediction. When coupled with iterative optimization methods, they require massive computational effort on supercomputers. The problem formulation process requires identification of design variables, a cost function and constraints on the system for its safe performance. Depending on the class of problems and needs, several types of design variables and objective functions can be identified. Constraints usually involve physical limitations, material failure, buckling load, vibration frequencies and other such response quantities. These problems can be transcribed into a *standard NLP model* defined as:

**Problem P.** Find a design variable vector $\boldsymbol{x}$ of dimension $n$ to minimize a cost function $f(\boldsymbol{x})$ with $\boldsymbol{x}$ in the feasible set

$$S = \{\boldsymbol{x}:\ g_j(\boldsymbol{x}) = 0,\ j = 1 \text{ to } p;\ g_j(\boldsymbol{x}) \le 0,\ j = p+1 \text{ to } m;\ x_{i\mathrm{l}} \le x_i \le x_{i\mathrm{u}}\} \tag{1}$$

where $x_{i\mathrm{l}}$ and $x_{i\mathrm{u}}$ are lower and upper bounds on the $i$th design variables.

The simple design variable bound constraints are easy to impose. Therefore they are treated separately in numerical algorithms to effect efficiency. It is important to highlight *features* of the foregoing NLP model:

(1) The model is applicable to all problems with continuous design variables. *Multiobjective and discrete variable* problems can also be treated after certain extensions.

(2) The functions of the problem are assumed to be *continuous and differentiable*. Problems having *nondifferentiable functions* can be treated with additional computational effort [39]. Also, the gradients of *active constraints* are assumed to be linearly independent at the optimum.

(3) The cost and *constraint functions are usually implicit* as well as explicit functions of design variables. They can, however, be calculated once a design is specified.

(4) *Gradients* of the functions are needed in numerical methods of optimization. Efficient methods to calculate them, taking advantage of the structure of engineering design problems, have been developed [9,10,15,24,30,43,49–51].

(5) Realistic analysis models can be quite large requiring enormous calculation to evaluate implicit functions at a trial design. Thus, the *number of calls for function evaluations* is a measure of the efficiency of an optimization algorithm.

(6) Engineering design problems usually have a large number of inequality constraints. However, the majority of them are not binding at the optimum point. Since the set of binding constraints is not known a priori, it must be determined in the solution process. Design sensitivity analysis (gradient evaluation) for problems involving implicit functions needs large computation, so a *potential constraint strategy*, is which only a subset of all the inequality constraints is differentiated in each iteration, should be used. The number of gradient evaluations at each iteration is, therefore, another measure of efficiency.

## 3. GENERAL CONCEPTS

In this section some general concepts are discussed based on which several computational algorithms for the design optimization have been developed. Most algorithms are based on the general iterative equation

$$\boldsymbol{x}^{(j+1)} = \boldsymbol{x}^{(j)} + \alpha_j \boldsymbol{d}^{(j)}, \quad j = 0, 1, 2, \ldots \tag{2}$$

where the superscript $j$ represents the iteration number, $\boldsymbol{x}^{(j)}$ is an estimate of the optimum point, $\alpha_j \boldsymbol{d}^{(j)}$ is a change in design, $\alpha_j$ is the step size, $\boldsymbol{d}^{(j)}$ is a search direction calculated using function values and their gradients, and $\boldsymbol{x}^{(0)}$ is an initial design estimate given by the designer. Optimization algorithms are broadly classified as primal and transformation methods.

### 3.1. Primal methods

In primal methods the direction vector $\boldsymbol{d}^{(j)}$ is calculated using the function values and their gradients at the point $\boldsymbol{x}^{(j)}$. The step size calculation needs only the function values. Different algorithms can be generated depending on how the direction $\boldsymbol{d}$ and step size $\alpha$ are calculated. In many algorithms, $\boldsymbol{d}$ is calculated by solving a linear or quadratic programming subproblem. For example, a linear programming (LP) subproblem is defined by linearizing the cost and constraint functions at the $j$th iteration as follows:

**Subproblem LP.** Minimize $(\boldsymbol{c}, \boldsymbol{d})$ subject to the linearized constraints

$$g_i(\boldsymbol{x}^{(j)}) + (\boldsymbol{G}^i, \boldsymbol{d}) = 0 \quad \text{for all } i \in I_1 \tag{3}$$

$$g_i(\boldsymbol{x}^{(j)}) + (\boldsymbol{G}^i, \boldsymbol{d}) \leq 0 \quad \text{for all } i \in I_2 \tag{4}$$

$$|d_l| \leq \xi_l; \quad l = 1 \text{ to } n \tag{5}$$

where $(z, y)$ implies $z^{\mathrm{T}}y$, $\boldsymbol{c}$ is the gradient of the cost function, $\boldsymbol{G}^i$ is the gradient of the $i$th constraint function, $\xi_l$ is a limit on the change in the $l$th design variable (called the *move limit*), and $I_1$ and $I_2$ are the *potential constraint index sets* for the equality and inequality constraints, respectively.

There are several ways to define the sets $I_1$ and $I_2$ in eqns. (3) and (4), and each method gives a different search direction [48]. Any linear programming code can be used to solve Subproblem LP for the direction $\boldsymbol{d}$. In this approach, called *sequential linear programming* (SLP), the step size is taken as one. The approach is quite simple as it does not need development of a new computer program. Existing LP software can be used to solve problems. However, there are difficulties with the procedure in selecting proper move limits. Also, the method may fail to converge if the optimum solution is not at a vertex [7,30].

In order to develop better algorithms, the linear step size constraint in eqn. (5) is replaced [15,30] by a constraint on the length of vector $\boldsymbol{d}$ as $0.5(\boldsymbol{d}, \boldsymbol{Hd}) \leq \xi^2$, where $\boldsymbol{H}$ is a positive definite matrix and $\sqrt{2}\,\xi$ is the limit on the length of $\boldsymbol{d}$. $\boldsymbol{H}$ is taken as an identity matrix in some algorithms. The direction $\boldsymbol{d}$ obtained as a solution of Subproblem LP with the move limits of eqn. (5) replaced by the foregoing quadratic step size constraint, is the same as solution of the following QP subproblem [7,30,52]:

**Subproblem QP.** Minimize $(\boldsymbol{c}, \boldsymbol{d}) + 0.5(\boldsymbol{d}, \boldsymbol{Hd})$ subject to the linearized constraints of eqns. (3) and (4).

Several algorithms and software are available for solving Subproblem QP [16,30]. The primal methods differ in the way the LP or QP subproblems are defined, and the step size is calculated. For example, definitions of the index sets $I_1$ and $I_2$ in eqns. (3) and (4), and the selection of $\boldsymbol{H}$, $\xi$ and $\xi_l$ affect the direction of search as well as the rate of convergence of an algorithm [48]. Definition of the sets $I_1$ and $I_2$ is important for the efficiency of the algorithms. The algorithms that include all the constraints in the sets $I_1$ and $I_2$ at each iteration are not suitable for engineering design applications. This is due to the fact the gradients of all the constraints are calculated, which is quite expensive. In addition, the dimension of the subproblem becomes quite large. Therefore, it is important to use a potential constraint strategy in the algorithms.

The step size $\alpha$ in eqn. (2) can be determined by minimizing a descent function for the algorithm. In numerical calculations, only an approximate minimum along $\boldsymbol{d}$ is determined to effect efficiency. The *descent function* measures the progress of the interactive process toward the optimum and ensures that every new design is a better estimate than all the previous ones. It is constructed by adding a penalty term for the constraint violations to the cost function, as $\Phi(\boldsymbol{x}, r) = f(\boldsymbol{x}) + P[\boldsymbol{g}(\boldsymbol{x}), r]$, where $r$ is a scalar or vector of penalty parameters and $P$ is a real-valued function whose action of imposing a penalty is controlled by $r$. Computational algorithms based on the foregoing philosophy are called *descent methods*. They have a *global convergence property*, i.e. they converge to a local minimum starting from any point.

### 3.2. Transformation methods

These methods transform Problem P to a sequence of unconstrained optimization problems whose solutions converge to a solution of the original problem. The transformed function is similar to the function $\Phi(x, r)$ defined earlier. They include *barrier and penalty function methods* as well as the augmented Lagrangian or multiplier methods [53]. Details of the transformation methods can be found in Refs. [7,53,54] and the literature cited there. The idea of transformation methods is very attractive because existing unconstrained optimization algorithms can be used, so they are easy to program on the computer. In addition, it is shown in Ref. [53] that the computation of gradients of individual constraints is not required to calculate the gradient of the transformed function. This is a good feature for engineering design problems where calculation of gradients of individual constraints is quite expensive. In spite of these good features a computational study of various methods for static response structural design problems showed the transformation methods to be inefficient [53]. This is primarily due to the need for exact unconstrained minimization steps. In another recent study [54], the transformation methods were applied to dynamic response problems. For that class of problems they appear to be more suitable than the primal methods since the gradient of only one time-dependent functional is needed.

Since the transformed function $\Phi$ can be same as the descent function used in the primal methods, there is some commonality between the two approaches. Indeed, the two approaches can be merged and perhaps better algorithms can be developed. Further research is needed to exploit this idea in order to develop and evaluate new algorithms.

## 4. ATTRIBUTES OF A GOOD ALGORITHM

It is important to know the attributes of a good algorithm so that proper ones can be selected for practical applications. This is particularly true for complex and large-scale applications. Based on numerical experience, the attributes of a good algorithm are identified as follows [8,30]:

(1) *Reliability*. The algorithm must have proof of convergence to a local optimum point regardless of the starting design. Such algorithms are called *globally convergent, robust, reliable or stable*.

(2) *Generality*. Generality means that the algorithm is able to treat equality as well as inequality constraints and there is no restriction on the form of the function of the problem, such as linear, posinomial, separable, etc.

(3) *Efficiency*. The algorithm should converge in the least number of iterations and the number of calculations within one iteration should be low. The rate of convergence of an algorithm can be accelerated if second-order information is used. Within one iteration, the use of a potential constraint strategy to define the subproblem, efficient methods to solve it, and the minimum number of trials to calculate the step size reduce the number of calculations. Elimination of the line search can improve efficiency [55]. This may not be possible, however, because the progress of the algorithm towards a solution cannot be measured without a line search on a descent function.

(4) *Ease of use*. The algorithm must be easy to use by experienced as well as inexperienced designers. From a practical standpoint, this is an extremely important requirement. An algorithm requiring selection of tuning parameters will be difficult to use as knowledge and understanding

of the mathematical structure of the algorithm will be necessary. Such an algorithm is unsuitable for general usage.

## 5. PRIMAL METHODS BASED ON LINEAR INFORMATION

In this section several primal methods that use only first-order (gradient) information are briefly reviewed. Some of these methods have been investigated theoretically as well as numerically in Ref. [7] for structural design problems.

### 5.1. Feasible directions method

The idea behind the method [56] is to move from one feasible design to an improved feasible design while maintaining descent on the cost function. This is in contrast to some other methods which can also iterate through the infeasible region. In some applications, the system model employed may be simply invalid or meaningless outside the feasible region. The feasible directions method and other similar methods assume an important role in such applications. The method was developed for only inequality constrained problems ( $p$ is zero in Problem P). It solves an LP subproblem for an *improving feasible direction* – a direction which keeps the design feasible for a finite step along $\boldsymbol{d}$ as well as reduces the cost function, i.e. $(\boldsymbol{c}, \boldsymbol{d}) < 0$ and $(\boldsymbol{G}^i, \boldsymbol{d}) < 0$ for $i \in I_2$. The problem satisfying these conditions is defined as: minimize $\max\{(\boldsymbol{c}, \boldsymbol{d}); (\boldsymbol{G}^i, \boldsymbol{d})$ for all $i \in I_2\}$. The minimax problem can be converted to an LP subproblem for determining the search direction. After solving the direction-finding problem, a step size $\alpha_j$ is found such that $f(\boldsymbol{x}^{(j)} + \alpha \boldsymbol{d}^{(j)}) < f(\boldsymbol{x}^{(j)})$ and $g_i(\boldsymbol{x}^{(j)} + \alpha \boldsymbol{d}^{(j)}) \leq 0$ for all $i$. The method is fundamentally for inequality-constrained problems. Equality constraints can be treated by adding a penalty term to the cost function. It is, however, difficult to decide penalty parameters for the augmented cost function. Improper choice of parameters may impose a large penalty for violation of equality constraints and may completely dominate the real cost function, thus slowing down convergence to the optimum. Theoretically the feasible directions method is quite attractive and has been successfully used to solve many structural design problems [4,21,57]. Computationally, however, there are some difficulties. Firstly, a feasible starting point is needed which requires special algorithms to treat arbitrary starting points. Secondly, the algorithm requires specification of a parameter (called the push-off factor) for each constraint to compute the improving feasible direction. These push-off factors control the rate of convergence of the algorithm. It is difficult to select proper values for these factors for each problem without considerable numerical experimentation.

### 5.2. Gradient projection method

The gradient projection method *projects the gradient of the cost function* onto the constraint tangent hyperplane of the active constraints. It was developed by Rosen [58] who was motivated to develop a method in which an approximate search direction could be obtained in a closed form rather than by solving an LP or QP subproblem. The linearized inequality constraints in eqn. (4) are treated as equalities, and a closed-form solution for the search direction $\boldsymbol{d}$ is obtained by writing the Kuhn–Tucker optimality conditions. It is interesting to observe that the closed-form solution can be written as $\boldsymbol{d} = \boldsymbol{d}^1 + \boldsymbol{d}^2$, where $\boldsymbol{d}^1$ is the direction of descent for the

cost function and $\boldsymbol{d}^2$ is the direction of constraint correction normal to $\boldsymbol{d}^1$. An approximate step size calculation depending on a specified reduction in the cost function is given in Ref. [15]. Some properties of the vectors $\boldsymbol{d}^1$ and $\boldsymbol{d}^2$ that have useful graphical interpretation are also given there. One of the properties is that when $\boldsymbol{d}^1 \approx \boldsymbol{0}$, the Kuhn–Tucker optimality conditions for Problem P are satisfied.

The gradient projection method is quite simple to implement and use. However, there are serious numerical difficulties in actual applications. The major one is that it is difficult to impose global convergence. From a feasible point, an arbitrary step is taken in the $\boldsymbol{d}^1$ direction which decreases the cost function. Since $\boldsymbol{d}^1$ is tangent to the constraint surface, any finite step results in violation of the constraints. Therefore constraints must be corrected while keeping the cost function below its value at the previous feasible point. Several numerical procedures have been tried to accomplish this objective [7]. They are, however, quite cumbersome and inefficient.

### 5.3. Generalized reduced gradient method

The reduced gradient method was developed based on a simple variable elimination technique for equality-constrained problems [59]. Dependent and independent variables are identified in the linearized subproblem and the dependent variables are eliminated from it. This way the gradient of the cost function in terms of only the independent variables – called the *reduced gradient* – is obtained. The generalized reduced gradient (GRG) method is an extension of the reduced gradient method to accommodate nonlinear inequality constraints. In this method, a search direction is found such that for any small move the current active constraints remain precisely active. If some active contraints are not precisely satisfied due to nonlinearity of the constraint functions the Newton–Raphson method is used to return to the constraint boundary [60–62]. The GRG method can be considered somewhat similar to the gradient projection method because the reduced gradient is precisely the projected gradient direction $\boldsymbol{d}^1$.

The GRG algorithm is inefficient due to the Newton–Raphson (NR) iterations during line search. The gradients of the constraints need to be recalculated and a Jacobian matrix needs to be inverted at every NR iteration during the line search. To overcome this inefficiency, many numerical schemes have been used, e.g. the use of a quasi-Newton formula to update the Jacobian without recomputing the gradients. This can cause problems if the set of independent variables changes every iteration. Another difficulty is to select a feasible starting point. Special algorithms must be used to handle arbitrary starting points. The method is not suitable for large-scale problems when inequalities are converted to equalities by adding slack variables because the number of constraints needing gradient evaluation increases enormously. If a potential constraint strategy is used, the method becomes essentially the same as the gradient projection method [7].

Recently, it has been proposed to combine the feasible directions and the reduced gradient methods [57]. The equality constraints are treated using the variable elimination scheme of the reduced gradient method. The inequalities are treated as in the feasible direction method. This new algorithm philosophically combines the good features of the two algorithms. However, the numerical difficulties of the two algorithms, as discussed previously, still remain.

### 5.4. Linearization method

A linearization method whose global convergence has been proved with a unique potential constraint strategy was developed in the 1970s [63]. This unique potential constraint strategy is

highly suitable for large-scale applications [52,64]. In the definition of Subproblem QP in this method, the matrix $\boldsymbol{H}$ is taken as $\boldsymbol{I}$, and the index sets $I_1$ and $I_2$ are defined as $I_1 = \{i: \mid g_i(\boldsymbol{x}) \mid + \epsilon(\boldsymbol{x}) \geq 0, \ i = 1 \text{ to } p\}$ and $I_2 = \{i: \ g_i(\boldsymbol{x}) + \epsilon(\boldsymbol{x}) \geq 0, \ i = p+1 \text{ to } m\}$. The constraint-checking parameter, $\epsilon(\boldsymbol{x})$, is defined using any specified positive constraint $\delta$ as $\epsilon(\boldsymbol{x}) = \delta - F(\boldsymbol{x})$, where $F(\boldsymbol{x}) \geq 0$ is the maximum constraint violation. Note that some of the violated equality and inequality constraints may not be included in the definition of the QP subproblem due to the negative value of $\epsilon(\boldsymbol{x})$. It can be shown that $\| \boldsymbol{d} \| = 0$ is both necessary and sufficient to satisfy the Kuhn–Tucker necessary conditions for a local minimum of Problem P [63]. The step size $\alpha$ is determined by minimizing the descent function $\Phi(\boldsymbol{x})$, defined as $\Phi(\boldsymbol{x}) = f(\boldsymbol{x}) + RF(\boldsymbol{x})$, where the penalty parameter $R$ is selected to be larger than the sum of all the Lagrange multipliers. In practice, only an inexact line search is used to determine the step size $\alpha$. Recently, the method has been substantially enhanced [48] as described in Section 6.

### 5.5. Other methods

Several other methods have been developed and used to solve various problems [7,35,36,38]. These methods are useful because they are usually more efficient for a specific application. They can also work well in an interactive environment. Usually, near-optimum design can be obtained in just a few iterations. However, they usually lack in convergence to a precise optimum. Two such algorithms are briefly described.

#### 5.5.1. Cost function bounding algorithm

This algorithm can provide the capability for useful interactive queries during the iterative design optimization process. The basic purpose of the algorithm is to eliminate the expensive line search during the design updating phase [55]. The basic concept used is to determine the upper and lower bounds on the optimum cost and then search the design space between them to improve the bounds until an optimum is reached. Four subproblems that are variations of Subproblem QP are defined to conduct the search in an organized way [30]. Depending on various conditions of the design during an iteration, one or at the most two subproblems are solved to update the design.

#### 5.5.2. Method of moving asymptotes

A common step in all the primal methods analyzed so far is to linearize the coast and constraint functions with respect to the design variables to define a subproblem. Unlike the other primal methods, the key idea of the method of moving asymptotes is to linearize each function of the problem with respect to some intermediate variables [65]. For each function, the sign of its first partial derivative with respect to each design variable is checked. Depending on whether the sign is positive or negative, intermediate variables are defined as $z_i = 1/(U_i - x_i)$ or $z_i = 1/(x_i - L_i)$ where $U_i$ and $L_i$ are specified numbers that need to be adjusted every iteration. The method is a generalization of the procedure suggested by Starnes and Haftka [66]. It has been applied to small-scale problems having only inequality constraints.

## 6. PRIMAL METHODS BASED ON QUADRATIC INFORMATION

An idea to extend Newton's method for unconstrained problems to solve general constrained problems was proposed by Wilson [67]. The idea was to calculate the Hessian of the Lagrange

function and use it in the search direction determination. The QP subproblem could be solved for the search direction where $\boldsymbol{H}$ was the Hessian of the Lagrange function. The method needed estimates of the Lagrange multipliers to define the Lagrange function, whose second derivatives were also needed. These were the major difficulties and the method could not be used for practical applications until it was proposed to approximate the Hessian using its gradients [68]. The method was further developed and established on sound foundations by Han [69] and Powell [70]. In the method, the Hessian of the Lagrange function is approximated using the change in design, $\alpha \boldsymbol{d}$, and the gradient of the Lagrangian at two successive iterations. It is required to be symmetric and positive definite throughout the iterative process. Several procedures can be used for updating the Hessian [16]. Extensive numerical experimentation with a program based on the Wilson–Han–Powell algorithm has been done in Ref. [71]. In Ref. [72] another FORTRAN subroutine called VMCON, based on the same algorithm, has been developed. Another subroutine, called NPSOL, that is implemented with different numerical schemes, has been developed in Refs. [16, 73]. The methods have been named variously as the recursive quadratic programming (RQP), projected Lagrangian, successive quadratic programming (SQP), or constrained variable metric (CVM) methods. An important feature of the methods is that the sequence of designs $\{\boldsymbol{x}^{(j)}\}$ generally exhibits a local superlinear rate of convergence.

### 6.1. Method without a potential constraint strategy

The Wilson–Han–Powell SQP algorithm does not implement a potential constraint strategy. Thus, all constraints are included in Subproblem QP. The subproblem may be incompatible due to the inconsistency of the linearized constraints. To overcome this difficulty [70], a dummy variable $c$ is used to express the linearized inequality constraints in eqn. (3) in the form $cg_i(\boldsymbol{b}^{(j)}) + (\boldsymbol{G}^i, \boldsymbol{d}) \leq 0$, where $0 < c \leq 1$. The objective function of Subproblem QP is modified by adding a term $(1 - c)M$, where $M$ is a large positive constant. If $c = 1$, the original QP subproblem is obtained. If the QP subproblem has no solution, then $c < 1$ allows additional freedom to find a direction vector to the modified subproblem. The step size along the direction is determined by minimizing the descent function. Extensive numerical experimentation with the algorithm along with other competitive NLP algorithms has been done in Ref. [74]. It has been concluded that the CVM method is the most efficient and reliable. The method is, however, not suitable for large-scale applications unless a potential constraint strategy is used, which is discussed in the sequel.

### 6.2. Method with a potential constraint strategy

Recently, the CVM method has been successfully implemented with a potential constraint strategy [75–77]. It was observed in Section 5.4 that the linearization method is highly suitable for engineering design applications because it uses a good potential constraint strategy. However, it can be slow because only linear information is used. Therefore several efficient numerical schemes have been developed to enhance the method [75–77]. One of the major enhancements is to generate an approximate Hessian $\boldsymbol{H}$ of the Lagrange function and use it in the definition of the QP subproblem. This modification can give a local superlinear rate of convergence to the method with a potential constraint strategy. The algorithm, called PLBA (Pshenichny–Lim–Belegundu–Arora), essentially follows the steps of the linearization method except that Hessian

updating is used. Some computational modifications related to step size determination have been also found to be useful. These are (i) the step size should be allowed to be larger than one, and (ii) polynomial interpolation during line search can reduce the number of function evaluations and thus improve efficiency. A difficulty encountered with the new algorithm was that the updated Hessian matrix could deteriorate with potential constraint strategy. For a small test problem, the algorithm with Hessian updating actually took more iterations to converge compared to the linearization method. The difficulty was overcome by devising re-start procedures. The basic idea was that if the Hessian became ill-conditioned, or the direction of descent could not be found, then it was re-set to identity. This numerical procedure has worked quite well.

The PLBA algorithm and its variations have been extensively applied to static as well as dynamic response problems with success [75–77]. The performance of the CVM methods with and without a potential constraint strategy has been evaluated [78] on NLP problems given in Ref. [74]. The performance of the method has also been investigated on structural design problems [79]. The algorithm has been recently further enhanced and extensively evaluated [48,49]. Successful extensions of the linearization method have been also recently reported in Ref. [80]. The PLBA algorithm has proved to be quite reliable and accurate. It is therefore recommended for general applications. Some ideas to improve the efficiency of the method for very large-scale problems are discussed later in the paper.

### 6.3. Hybrid methods

The CVM algorithms can iterate through the infeasible region of the design space. The initial design estimate can be arbitrary, and if it is too optimistic, the number of constraint violations can be large. Since the potential constraints can be quite different at two consecutive iterations, incorrect second-order information may be accumulated in the Hessian matrix. This may give bad directions of search, thus destroying the superlinear rate of convergence. This behavior was observed, as noted in the previous section, when the method was applied to a small-scale problem. The foregoing discussion indicates that it may be prudent to use an algorithm based only on linear information in the initial stages and thus not accumulate any second-order information. After a few iterations, once the initial design has improved and the feasible point has been found, a switch can be made to the CVM method. Such a two-phase algorithm, which uses the cost function bounding algorithm [55] initially followed by the PLBA algorithm, has been developed [75]. All these algorithms are available in a computer-aided design system called IDESIGN [81]. It has been concluded that the hybrid methods can be efficient compared to the CVM algorithms.

## 7. OPTIMALITY CRITERIA METHODS

A philosophically different approach has been extensively followed for structural design optimization [82]. In this approach, known as the optimality criteria methods, necessary conditions for the optimality of the constrained model are written. These conditions are generally nonlinear. For highly idealized and simple models, they lead to analytical expressions that can be given physical interpretations. In more complicated cases, numerical methods must be used to solve the nonlinear necessary conditions. The numerical solutions can also lead to significant

qualitative results. The approach has been used to gain insights into the optimum behavior of systems. A comprehensive review of the subject is given in Refs. [5,82]. It has been also recently shown [7,30] that *Newton's iteration* to solve the nonlinear set of necessary conditions is equivalent to solving a quadratic programming subproblem which is used in many nonlinear programming methods described previously. In Ref. [83] the equivalence of the numerical optimality criteria and gradient projection methods has been shown. It appears that the two philosophically different approaches merge when numerical methods are used to solve optimality conditions.

## 8. RECENT DEVELOPMENTS

Considerable work continues to be done to develop the computational design optimization field so that the new technology can be used in actual design practice. In this section we briefly review some of the recent developments in the field that contribute to the achievement of this goal.

### 8.1. Interactive use of optimization

Interactive design optimization is a software environment that allows the designer to interact with the iterative design process. Such an environment is possible when interactive optimization algorithms are implemented into a software that also has commands for *design decision-making*. Some algorithms suitable for interactive design optimization have been developed recently [30]. These have been implemented into the IDESIGN system to create an interactive environment [81,84,85]. The system has been used for static [86] as well as dynamic response applications [87]. It has been found to be useful for practical applications.

The optimal design of large complex systems may need huge computer time. In such cases it is useful and important to monitor the optimal design process interactively. The histories of the cost function, the constraint functions, the design variables, the maximum violation, and the convergence parameter can be monitored. If the design process is not proceeding satisfactorily, which could be due to inaccuracies or errors in the problem formulation, it is necessary to stop the process and check the problem formulation. This saves human as well as computer resources. The relative sensitivity of problem functions to various design variables can be monitored. Some variables may have little effect compared to others, so they can be dropped from the formulation. If there are large violations of the constraints, it may be useful to know whether a feasible design can be obtained and what the penalty on the cost function would be. More details of all the capabilities and their use can be found in Refs. [30,81,84–87]. The creation of an interactive design environment can be facilitated with the use of modern knowledge engineering and expert system techniques. These aspects are discussed in Refs. [81,84].

### 8.2. Development of algorithms

The optimal design of large complex systems needs *robust and efficient algorithms*. Work on this aspect continues and new algorithms continue to be developed and tested. The recently developed sequential quadratic programming algorithm with potential constraint strategy [48,75,76] has proved to be quite robust for a wide range of applications. However, for large-scale

applications the algorithm becomes inefficient due to the large dimension of the approximate Hessian. The matrix is assumed to be dense, so the number of operations with its use become quite large. Efforts are underway to exploit the sparsity of the matrix, and even use a diagonal form for it. With these extensions the method is expected to become quite efficient as well as robust for large-scale applications.

Another algorithm, called generalized geometric programming (GGP), has recently been used for shape optimization problems [88]. That algorithm may have potential for other practical applications, and should be investigated.

The transformation methods described in Section 3 have been re-evaluated in Ref. [54] for dynamic response applications. They have been found to be superior to the primal methods for this class of problems. The methods need to be reexamined to see if their numerical performance can be improved. The methods are quite attractive because they are easy to understand and implement on the computer. A micro-analysis of the algorithms is being performed relative to large-scale design problems. The idea is to determine inefficient steps and improve them. The algorithms are believed to have good potential for large-scale applications.

Another area of active research is the development of algorithms suitable for parallel processing [39]. Such algorithms can reduce computer time by orders of magnitude. This will make optimization techniques suitable for very large-scale applications.

### 8.3. Computer implementation of algorithms

Computational algorithms for design optimization must be properly implemented on the computer for use in practical applications. Transcribing an algorithm into a computer program is an *art* to some degree. A theoretically convergent algorithm may be implemented badly such that it is neither convergent nor reliable, and vice versa. The robust implementation of an algorithm requires considerable experience and insight, and it is best to leave the task to the specialist. Proper implementation of algorithms requires a certain amount of heuristics which can be developed only with experience and by solving many problems of varying difficulty. An excellent discussion on this subject is given in Refs. [16,48]. A detailed discussion of these aspects from an artificial intelligence point of view is contained in Ref. [84]. Each optimization algorithm has certain parameters that must be selected every time a problem is solved. Specification of these parameters can have considerable influence on the performance of the algorithm. This has been amply demonstrated in Refs. [48,49,76] in a detailed study of the PLBA algorithm. A good optimization algorithm should be relatively insensitive to variations in the parameters. As a minimum requirement, the algorithm must converge with every specification of the parameters within an allowable range. These aspects and attributes, discussed in Section 4, can be used in selecting algorithms and the associated software for practical applications.

Good software is essentail for design optimization of large complex systems. This has generally been a stumbling block in practical applications of optimization. Recently, some general purpose software has become available [36,39,42,81]. Efforts are also underway to incorporate optimal design capability into general purpose finite element programs. Undoubtedly, more effort is needed in this area to make optimization methodology available to designers. This is particularly true for complex system that require multidisciplinary capabilities. A software system must be highly sophisticated and modular allowing it to be upgraded and expanded. Modern *object-oriented programming* concepts, database design techniques, and *database management systems* (DBMS) can be quite useful in this regard. The importance of using

central databases has been realized and a database design methodology has been developed for structural analysis and design optimization [89]. Various DBMS suitable for engineering applications have been reviewed [90]. Based on that study, the requirements of a good DBMS for engineering applications were developed. A suitable DBMS was implemented and evaluated. Based on that evaluation, an improved DBMS has been developed [91]. Design optimization programs that use the DBMS have been designed, implemented, and evaluated [84–86,89,91]. It appears that the use of a DBMS is of critical importance and its use will increase in the future as more multidisciplinary design problems are treated in an integrated manner.

### 8.4. Evaluation of gradients of implicit functions

The problem of calculating gradients of implicit functions has been called *design sensitivity analysis* [15,50,51]. Methods of such analysis for linear systems are reviewed in Ref. [10]. The methods have been extensively used and evaluated [15,24,43,50,51]. Design sensitivity analysis methods for dynamics and controls of systems [49,92,93] and nonlinear structures [94,99] have been developed. Path-dependent response problems have been investigated [100,101]. Recent developments in these important areas have been reviewed in Ref. [102]. These are active areas of research and more work needs to be done to completely develop them so that complex and large-scale systems can be optimized. The design sensitivity analysis methods can be used not only in optimization algorithms but also for reliability-based design and analysis of systems [103–105].

## 9. CONCLUDING REMARKS

A review of the computational design optimization field is presented relative to the structural design problem. General *concepts of computational algorithms* are explained and the definition of a good algorithm is given. Some computational algorithms based only on linear information that have been used successfully in the past are described. These algorithms have been superceded recently by the methods that generate and use second-order information. An algorithm belonging to the latter class that is suitable for engineering applications is described. The *interactive* use of optimization is discussed, and it is noted that such use can be beneficial in practical design. Current thinking on the development of algorithms for *large-scale* and *complex applications* is described. Proper *implementation* of computational algorithms into a software is also discussed.

Based on experience with several algorithms, it is suggested that designers who are not experts in optimization should use only *general and reliable* methods. Although such methods require slightly more computational effort compared to other algorithms, they are more *cost effective* in the long run. Efficiency, although important, can be sacrificed for reliability. A few more CPU cycles of the machine are worth the wait if the algorithm can reliably give an optimum solution. The use of unreliable and ad hoc methods and software based on them can actually be more costly and frustrating in terms of a designer's time and uncertainty about the solution. There are many numerical difficulties in solving large complex problems. They need not be compounded by the use of unreliable algorithms and programs based on them.

The transcription of an algorithm into good software requires considerable experience, time and resources [48]. This should be a task for optimization experts and computer scientists. Design engineers need not be involved in algorithm development or its implementation. Their

job should be to properly formulate the design optimization problem [30], solve it using an available software, and interpret the results. They should concentrate on investigating other conceptual designs or formulations for their application. Once a concept is properly formulated, a well developed and robust software can solve the problem very quickly. This way optimization methods and programs can be used as tools in the creative process of designing systems.

## 10. FUTURE DIRECTIONS

Based on the review of CDO field, the following areas are identified for further research:

(1) *Efficient algorithms*. Research should be continued in developing more efficient, robust and general algorithms. The SQP method has great promise in the regard. However, better updating procedures for Hessians, exploitation of sparsity in the matrix and better step size determination procedures need to be developed. This is especially true for large-scale applications where the number of design variables is large. The concepts of scaling the cost function and/or constraints need to be developed so that the subproblem is well scaled and also the step size determination process is well defined. If the cost function is not properly scaled, the step size can be either too large or too small which is not desirable. The *scaling concepts* can have a tremendous effect on efficiency and overall performance of the algorithm [30]. Transformation of the design variables to ensure a better scaled problem is another area that needs attention. The augmented Lagrangian methods have much promise for the success on large-scale engineering design problems. They need to be exploited and further developed for engineering applications. Merging of the good features of the augmented Lagrangian and SQP methods needs to be investigated.

(2) *Parallel algorithms*. The need to develop parallel algorithms is obvious. As more parallel processing machines become commonplace and system software becomes more sophisticated, the use of such machines will grow. Therefore we shall need parallel algorithms for design optimization, especially for large-scale systems.

(3) *Advanced algorithms and applications*. Several applications have functions that cannot be differentiated, so computational algorithms for *nondifferentiable* problems need to be developed. In addition, some problems require *global* optimum solution, so methods to treat such problems need to be developed. In the past, the development of computational algorithms for these complex problems has been hindered due to unavailability of fast computers. However, with the availability of supercomputers and parallel processors, we can begin to develop computational strategies for such complex applications.

(4) *Software development*. The availability of good software is a key to the successful use of design optimization technology. More attention needs to be paid in developing software that is easily maintainable and extendable. Object-oriented program development and the use of good database management systems can be quite useful in this regard. The use of knowledge-based expert system development techniques can be useful in developing robust programs and good user interfaces.

(5) *Large-scale applications*. The design of realistic systems can involve cooperation from different disciplines of engineering, e.g. integration of structural analysis, thermal analysis, controls and reliability-based design fields. This *interdisciplinary* nature of the design problems inherently leads to large-scale problems. It is important to properly integrate different disciplines, so more attention needs to be given to this problem. The scaling concepts and design variable transformations mentioned earlier can offer advantage in dealing with different types of

design variables encountered in such applications. Many of the applications have multiple objective functions, so computational algorithms for these problems need to be developed [22]. One of the prime areas for application of computational design optimization techniques is the reliability-based design. This area needs to be pursued vigorously.

## ACKNOWLEDGEMENT

The work reported in this paper is based on a part of the research sponsored by the U.S. National Science Foundation under the project,"Design Sensitivity Analysis and Optimization of Nonlinear Response of Structural Systems", Grant No. MSM 86-13314.

## REFERENCES

1 V.B. Venkayya, Structural optimization: A review and some recommendations, Int. J. Numer. Methods in Eng., 13 (1978) 205–228.
2 L.A. Schmit, Structural synthesis – its genesis and development, AIAA J., 19 (1981) 1249–1263.
3 H. Ashley, On making things the best – aeronautical uses of optimization, J. Aircraft, 19 (1982) 5–28.
4 G.N. Vanderplaats, Structural optimization – past, present and future, AIAA J., 20 (1982) 992–1000.
5 N. Olhoff and J.E. Taylor, On structural optimization, J. Appl. Mech., 50 (1983) 1139–1151.
6 J.S. Arora and A.D. Belegundu, Structural optimization by mathematical programming methods, AIAA J., 22 (1984) 854–856.
7 A.D. Belegundu and J.S. Arora, A study of mathematical programming methods for structural optimization. Part I: theory; part II: numerical aspects, Int. J. Numer. Methods in Eng., 21 (1985) 1583–1623.
8 J.S. Arora and P.B. Thanedar, Computational methods for optimum design of large complex systems, Comput. Mech., 1 (1986) 221–242.
9 R.T. Haftka and R.V. Grandhi, Structural shape optimization – a survey, Comput. Methods Appl. Mech. Eng., 57 (1986) 91–106.
10 H.M. Adelman and R.T. Haftka, Sensitivity analysis for discrete structural systems, AIAA J., 24 (5) (1986) 823–832.
11 R. Levy and O.E. Lev, Recent developments in structural optimization, J. Struct. Eng., ASCE, 113 (9) (1987) 1939–1962.
12 D.M. Himmelblau, Applied Nonlinear Programming, McGraw-Hill Book Co., New York, 1972.
13 W.R. Spillers, Iterative Structural Design, North-Holland, Amsterdam, 1975.
14 D.J. Wilde, Globally Optimal Design, Wiley, New York, 1978.
15 E.J. Haug and J.S. Arora, Applied Optimal Design: Mechanical and Structural Systems, Wiley–Interscience, New York, 1979.
16 P.E. Gill, W. Murray and M.H. Wright, Practical Optimization, Academic Press, New York, 1981.
17 U. Kirsch, Optimum Structural Design, McGraw-Hill, New York, 1981.
18 N.G.R. Iyengar and S.K. Gupta, Programming Methods in Structural Design, Wiley and Sons, New York, 1980.
19 G.V. Reklaitis, A.Ravindran and K.M. Ragsdell, Engineering Optimization, Wiley–Interscience, New York, 1983.
20 S.S. Rao, Optimization: Theory and Applications, 2nd edn., Wiley, New York, 1984.
21 G.N. Vanderplaats, Numerical Optimization Techniques for Engineering Design, McGraw-Hill, New York, 1984.
22 A. Osyczka, Multicriterion Optimization in Engineering, Halsted Press, Wiley, New York, 1984.
23 R.T. Haftka and M.P. Kamat, Elements of Structural Optimization, Martinus Nijhoff, Amsterdam, 1985.
24 E.J. Haug, K.K. Choi and V. Komkov, Design Sensitivity Analysis of Structural Systems, Academic Press, New York, 1986.
25 T.E. Shoup and F. Mistree, Optimization Methods with Applications for Personal Computers, Prentice-Hall Inc., Englewood Cliffs, NJ, 1987.
26 P.Y. Papalambros and D.J. Wilde, Principles of Optimal Design: Modeling and Computation, Cambridge University Press, New York, 1988.

27 A. Gajewski and M. Zyczkowski, Optimal Structural Design Under Stability Constraints, Kluwer Academic Publishers, Hingham, MA, 1988.
28 G.I.N. Rozvany, Structural Design via Optimality Criteria, Kluwer Academic Publishers, Hingham, MA, 1988.
29 J. Haslinger and P. Neittaanmäki, Finite Element Approximation for Optimal Shape Design: Theory and Applications, Wiley and Sons, New York, 1988.
30 J.S. Arora, Introduction to Optimum Design, McGraw-Hill Book Co., New York, 1989.
31 R.H. Gallagher and O.C. Zienkiewicz (Eds.), Optimum Structural Design: Theory and Applications, Wiley, New York, 1973.
32 E.J. Haug and J. Cea (Eds.), Optimization of Distributed Parameter Structures, Vols. 1 and 2, Proc. NATO ASI Meeting, Iowa City (1980), Sijthoff and Noordhoff International Publishers B.V., Alphen aan de Rijn, The Netherlands, Rockville, MD, 1981.
33 O.E. Lev (Ed.), Structural Optimization: Recent Developments and Applications, Am. Soc. Civ. Eng., NY, 1981.
34 A.J. Morris (Ed.), Foundations of Structural Optimization, Proc. of NATO ASI Meeting, Liege, Belgium (1980), Wiley–Interscience, New York, 1982.
35 H. Eschenauer and N. Olhoff (Eds.), Optimization Methods in Structural Design, Proc. Euromech Colloquium 164, Siegen, FRG, 1982, Bibliographisches Institut, Zürich, 1983.
36 E. Atrek, R.H. Gallagher, K.M. Ragsdell and O.C. Zienkiewicz (Eds.), New Directions in Optimum Structural Design, Proc. Int. Symp. Optimum Structural Design, University of Arizona, Tucson, AZ (1981), Wiley, New York, 1984.
37 J. Sobieski (Ed.), Recent Experiences in Multidisciplinary Analysis and Optimization, NASA Conf. Publ. 2327, Proc. Symp. held at NASA Langley Research Center, Hampton, VA, 1984.
38 K.M. Ragsdell (Ed.), Proceedings of the CAD/CAM Robotics and Automation International Conference, University of Arizona, Tucson, AZ, 1985.
39 K. Schittkowski (Ed.), Computational Mathematical Programming, Proc. of NATO ASI, Bad Windsheim, FRG, July 23–August 2 (1984), Springer, Berlin, Heidelberg, New York, Tokyo, 1985.
40 J.A. Gero (Ed.), Design Optimization, Academic Press, Orlando, FL, 1985.
41 J.A. Bennett and M.E. Botkin (Eds.), The Optimum Shape: Automated Structural Design, Proc. Int. Symp., GM Research Laboratories, Warren, MI, September 30–October 1, 1985, Plenum Press, New York, 1986.
42 C.A. Mota Soares (Ed.), Computer Aided Optimal Design: Structural and Mechanical Systems, Proc. of the NATO Advanced Study Institute held in Troia, Portugal, June 29–July 11, 1986, Series F: Computer and System Sciences, Vol. 27, Springer-Verlag, New York, 1987.
43 H.M. Adelman and R.T. Haftka (Eds.), Sensitivity Analysis in Engineering, NASA CP 2457, 1987.
44 G.I.N. Rozvany and B.L. Karihaloo (Eds.), Structural Optimization, Proceedings of the IUTAM Symposium on Structural Optimization, Melbourne, Australia, February, 1988.
45 A.M. Brandt (Ed.), Foundations of Optimum Design in Civil Engineering, Kluwer Academic Publishers, Hingham, MA, 1988.
46 G. Fu and D.M. Frangopol, Multicriterion reliability-based optimization of structural systems, in: P.D. Spanos (Ed.), Probabilistic Methods in Civil Engineering, ASCE, New York, 1988, pp. 177–180.
47 D.M. Frangopol, Structural optimization using reliability concepts in design, J. Struct. Eng., ASCE, 111 (11) (1985).
48 C.H. Tseng and J.S. Arora, On implementations of computational algorithms for optimum design, Int. J. Numer. Methods in Eng., 26 (6) (1988) 1365–1402.
49 C.H. Tseng and J.S. Arora, Optimum design of systems for dynamics and controls using sequential quadratic programming, AIAA J. (to appear 1989).
50 J.S. Arora and A.K. Govil, Design sensitivity analysis with substructuring, J. Eng. Mech. Div., ASCE, 103 (1977) 537–548.
51 J.S. Arora and E.J. Haug, Methods of design sensitivity analysis in structural optimization, AIAA J., 17 (1979) 970–974.
52 A.D. Belegundu and J.S. Arora, A recursive quadratic programming method with active set strategy for optimal design, Int. J. Numer. Methods in Eng., 20 (1984) 803–816.
53 A.D. Belegundu and J.S. Arora, A computational study of transformation methods for optimal design, AIAA J., 22 (1984) 535–542.
54 J.K. Paeng and J.S. Arora, Dynamic response optimization of mechanical systems with multiplier methods, J. Mech., Transm. Autom. Des., Trans. ASME, 111 (1) (1989) 73–80.

55 J.S. Arora, An algorithm for optimum structural design without line search, in: E. Atrek, R.H. Gallagher, K.M. Ragsdell and O.C. Zienkiewicz (Eds.), New Directions in Optimum Structural Design, Wiley, New York, 1984, Chapter 20.
56 G. Zoutendijk, Methods of Feasible Directions, Elsevier, Amsterdam, 1960.
57 G.N. Vanderplaats, An efficient feasible directions algorithm for design synthesis, AIAA J., 22 (11) (1984) 1633–1640.
58 J.B. Rosen, The gradient projection for nonlinear programming. Part II: nonlinear constraints, J. Soc. Ind. Appl. Math., 9 (1961) 514–532.
59 P. Wolfe, Methods of nonlinear programming, in: J. Abadie (Ed.), Nonlinear Programming, North-Holland, Amsterdam, 1970, Chapter 6.
60 J. Abadie, Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints, in: R. Fletcher (Ed.), Optimization, Academic Press, New York, 1969, pp. 37–47.
61 G.A. Gabriele and K.M. Ragsdell, The generalized reduced gradient method: a reliable tool for optimum design, ASME J., Eng. Ind. Ser. B, 99 (1977) 394–400.
62 G.A. Gabriele and K.M. Ragsdell, Large scale nonlinear programming using the generalized reduced gradient method, ASME J. Mech. Des., 102 (1980) 566–573.
63 B.N. Pshenichny and Y.M. Danilin, Numerical Methods in Extremal Problems, Mir Publishers, Moscow, 1978.
64 K.K. Choi, E.J. Haug, J.W. Hou and V.N. Sohoni, Pshenichny's linearization method for mechanical system optimization, J. Mech., Transm., Autom. Des., 105 (1983) 97–103.
65 K. Svanberg, The method of moving asymptotes – a new method for structural optimization, Int. J. Numer. Methods in Eng., 24 (1987) 359–373.
66 J.H. Starnes and R.T. Haftka, Preliminary design of composite wings for buckling, strength and displacement constraints, J. Aircraft, 16 (1979) 564–570.
67 R.B. Wilson, A simplical algorithm for concave programming, Ph.D. Dissertation, Harvard University, Cambridge, MA, 1963.
68 V.M. Garcia-Palomares and O.L. Mangasarian, Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problem, Math. Prog., 11 (1976) 1–13.
69 S.P. Han, A globally convergent method for nonlinear programming, J. Optimization Theory Appl., 22 (1977) 297–309.
70 M.J.D. Powell, Algorithms for nonlinear functions that use Lagrangian functions, Math. Prog., 14 (1978) 224–248.
71 K. Schittkowski, The nonlinear programming method of Wilson, Han and Powell with an augmented Lagrangian type line search function. Part 1: convergence analysis; part 2: an efficient implementation with least squares subproblems, Numer. Math., 38 (1981) 83–114; 115–127.
72 R.L. Crane, K.E. Hillstrom and M. Minkoff, Solution of the general nonlinear programming problem with subroutine VMCON, ANL-80-64, Argonne National Laboratory, Argonne, IL, 1980.
73 P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright, Model building and practical implementation aspects in nonlinear programming, in: K. Schittkowski (Ed.), Computational Mathematical Programming, Proc. NATO ASI, Bad Windsheim, Germany, July 23–August 2, 1984, Springer, Berlin, Heidelberg, New York, Tokyo, 1985, 209–248.
74 W. Hock and K. Schittkowski, Test examples for nonlinear programming codes, in: Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer, Berlin, Heidelberg, New York, 1981.
75 P.B. Thanedar, J.S. Arora and C.H. Tseng, An efficient hybrid optimization method and its role in computer aided design, Comput. Struct., 23 (3) (1986) 305–314.
76 O.K. Lim and J.S. Arora, An active set RQP algorithm for engineering design optimization, Comput. Methods Appl. Mech. Eng., 57 (1986) 51–65.
77 O.K. Lim and J.S. Arora, Dynamic response optimization using an active set RQP algorithm, Int. J. Numer. Methods in Eng., 24 (10) (1987) 1827–1840.
78 J.S. Arora and C.H. Tseng, An investigation of Pshenichny's recursive quadratic programming method for engineering optimization: A discussion, J. Mech., Transm. Autom. Des., Trans. ASME, 109 (2) (1987) 254–256.
79 P.B. Thanedar, J.S. Arora, C.H. Tseng, O.K. Lim and G.J. Park, Performance of some SQP algorithms on structural design problems, Int. J. Numer. Methods in Eng., 23 (12) (1986) 2187–2203.
80 T.J. Beltracchi and G.A. Gabriele, An investigation of Pshenichny's recursive quadratic programming method for engineering optimization, J. Mech., Transm. Autom. Des., Trans. ASME, 109 (2) (1987) 248–253.

81 J.S. Arora and C.H. Tseng, Interactive design optimization, Eng. Optimization, 13 (1988) 173–188.
82 L. Berke and N.S. Khot, Structural optimization using optimality criteria, in: C.A. Mota Soares (Ed.), Computer Aided Optimal Design: Structural and Mechanical Systems, Springer-Verlag, New York, 1987, pp. 271–312.
83 J.S. Arora, Analysis of optimality criteria and gradient projection methods for optimal structural design, Comput. Meth. Appl. Mech. Eng., 23 (1980) 185–213.
84 J.S. Arora and G. Baenziger, Use of artificial intelligence in design optimization, Comput. Meth. Appl. Mech. Eng., 54 (1986) 303–323.
85 G.J. Park and J.S. Arora, Role of database management in design optimization systems, J. Aircraft, 24 (11) (1987) 745–750.
86 S.S. Al-Saadoun and J.S. Arora, Interactive design optimization of framed structures, J. Comput. Civ. Eng., ASCE, 3 (1) (1989) 60–74.
87 C-H. Tseng and J.S. Arora, Interactive design optimization of dynamic response, submitted, September 1989.
88 S.A. Burns, Generalized geometric programming with many equality constraints, Int. J. Numer. Methods in Eng., 24 (1987) 725–741.
89 T. SreekantaMurthy and J.S. Arora, Database design methodology for structural analysis and design optimization, Eng. with Comput., 1 (1986) 149–160.
90 T. SreekantaMurthy and J.S. Arora, A survey of database management in engineering, Adv. Eng. Software, 7 (3) (1985) 126–132.
91 J.S. Arora and S. Mukhopadhyay, An integrated database management system for engineering applications based on an extended relational model, Eng. with Comput., 4 (1988) 65–73.
92 C.C. Hsieh and J.S. Arora, Design sensitivity analysis optimization of dynamic response, Comput. Methods Appl. Mech. Eng., 43 (1984) 195–219.
93 C.C. Hsieh and J.S. Arora, An efficient method for dynamic response optimization, AIAA J., 23 (1985) 1484–1486.
94 Y.S. Ryu, M. Haririan, C.C. Wu and J.S. Arora, Structural design sensitivity analysis of nonlinear response, Comput. Struct., 21 (1985) 245–255.
95 C.C. Wu and J.S. Arora, Design sensitivity analysis of nonlinear response using incremental procedure, AIAA J., 25 (8) (1987) 1118–1125.
96 C.C. Wu and J.S. Arora, Design sensitivity analysis of nonlinear buckling load, Comput. Mech., 3 (1) (1988) 129–140.
97 K.K. Choi and J.L.T. Santos, Design sensitivity analysis of nonlinear structural systems. Part 1: Theory, Int. J. Numer. Methods in Eng., 24 (1987) 2039–2055.
98 J.B. Cardoso and J.S. Arora, A variational method for design sensitivity analysis in nonlinear structural mechanics, AIAA J., 26 (5) (1988) 595–603.
99 J.S. Arora and J.B. Cardoso, A design sensitivity analysis principle and its implementation into ADINA, Comput. Struct., 32 (3/4) (1989).
100 J.J. Tsay and J.S. Arora, Nonlinear structural design sensitivity analysis with path-dependent response, Comput. Meth. Appl. Mech. Eng., to appear, 1990.
101 J.J. Tsay and J.S. Arora, Optimum design of nonlinear structures with path dependent response, in: Proc. 30th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Mobile, Alabama, USA, April 1989; to appear in Structural Optimization.
102 R.T. Haftka and H.M. Adelman, Recent development in structural sensitivity analysis, Struct. Optim., 1 (3) (1989) 137–152.
103 A. Der Kiureghian and J-B. Ke, The stochastic finite element method in structural reliability, Prob. Eng. Mech., 3 (2) (1988) 83–91.
104 P-L. Liu and A. Der Kiureghian, Reliability of geometrically nonlinear structures, in: P.D. Spanos (Ed.), Probabilistic Methods in Civil Engineering, ASCE, New York, 1988 pp. 164–168.
105 P-L. Liu and A. Der Kiureghian, Optimization algorithms for structural reliability, in: W.K. Liu, T. Belytschko, M. Lawrence and T. Cruse (Eds.), Computational Probabilistic Methods, AMD Vol. 93, ASME, New York, 1988, pp. 185–196.