AIAA JOURNAL
Vol. 38, No. 11, November 2000

Engineering Design Optimization Using Interior-Point Algorithms

S. S. Rao*
University of Miami, Coral Gables, Florida 33124-0624
and
E. L. Mulkay†
Exxon Production Research Company, Houston, Texas 77252-2189

The interior-point methods have emerged as highly efficient procedures for solving linear programming problems in recent years. The interior point methods have superior theoretical properties as well as observed computational advantages over simplex methods at solving large linear programming problems and are found to be immune to degeneracy. It is a common observation in literature that most nonlinear programming methods that work well for small and medium sized problems are not able to solve large-scale problems efficiently. It is to be expected that interior-point methods, when used in an adaptive sequential linear programming strategy, might prove to be a powerful engineering optimization tool. The development of an adaptive sequential linear programming algorithm, based on an infeasible primal-dual path-following interior-point algorithm and fuzzy heuristics, is considered in this work for the solution of large-scale engineering design optimization problems. Several numerical examples are considered to demonstrate the effectiveness and efficiency of the present method. The examples include a part stamping problem, a turbine rotor design problem, and an optimal control problem related to tidal power generation. The number of design variables and constraints range from 12 to 402 and 20 to 20,300, respectively, for the part stamping problem. The turbine rotor problem involves 905 variables and 1081 constraints, and the optimal control problem has 2002 variables and 1001 constraints. The numerical results clearly demonstrate the superiority of the interior-point methods compared with the well-known simplex-based linear solver in solving large-scale optimum design problems. The superior performance of the present method is shown in terms of both computational time and the ability to handle degenerate problems.

I. Introduction

In the 1990s, the linear programming (LP) world has undergone nothing short of a revolution. With the 1984 publication of his seminal paper announcing a polynomial-time algorithm for LP, Karmarkar[1] started a flurry of activity in the area of interior-point LP. Karmarkar's original method involved the use of a projective scaling that served to move the current solution point to the center of the polytope in order to allow "long" steps toward the optimum. Interior-point algorithms get their name because the search directions generated strike out into the interior of the polytope rather than skirting around the boundaries, as do their simplex-like cousins. Newspaper and magazine reports appeared shortly after Karmarkar's paper claiming that this new method was nearly 50 times as fast as the simplex method at solving certain problems.[2,3] Apart from these claims, within two years of the publication of Karmarkar's method, an implementation of an interior-point method known as the dual affine variant demonstrated superiority over the simplex method.[3] Although no computationally efficient version of Karmarkar's algorithm has ever been programmed,[4] the algorithm served as an inspiration to a whole new way of thinking about LP. Since then, several papers have been written regarding interior-point methods, and many studies now show that for large linear problems, interior-point methods do better than simplex methods. Furthermore, unlike simplex-based algorithms that have difficulty with degenerate problems, interior-point methods are immune to degeneracy.[5] Most researchers agree that the best LP interior-point methods, after a decade of growth and maturity, are the primal-dual path following methods.[5,6] One such method is used in this work to solve the linear subproblems formed in an adaptive sequential linear programming (SLP) strategy for solving nonlinear engineering optimization problems.

The implementation of an efficient interior-point algorithm as the LP solving module of an SLP algorithm is described in this work. The goal is to show that large-scale problems with hundreds or thousands of design variables and constraints can be solved effectively with this method. After presenting a brief description of the infeasible primal-dual path-following LP algorithm, the SLP algorithm used in this study is summarized. The effectiveness of the method is demonstrated through the numerical study of several large-scale engineering design problems.

II. Interior-Point Methods vs the Simplex Algorithm

Linear programming represents one of the most well-developed segments of the optimization community. Since the introduction of the simplex algorithm by Dantzig in the 1940s, many successful applications in engineering, economics, finance, and operations research have been solved using LP techniques. The basic properties of an LP problem are 1) a vector of real variables, whose optimal values are found by solving the problem, 2) a linear objective function, and 3) linear constraints, both inequalities and equalities.

A geometric interpretation of the simplex algorithm shows that the simplex algorithm solves an LP problem by solving for a series of vertex solutions, terminating when the optimal vertex solution is found. In other words, intermediate solutions of the simplex method represent points along the boundaries of the feasible space. One additional feature of this geometric interpretation is that because the simplex algorithm searches for vertex solutions, it has difficulty solving problems that are degenerate. Furthermore, if the simplex method is able to solve a degenerate problem (by employing one of a number of heuristics), it always stops at a vertex solution even if an entire edge or face of the feasible space is equally optimal.

Interior-point methods, as the name implies, do not employ the same "search the boundaries" strategy to locate the optimal value. Instead, a search direction is generated that strikes out into the interior of the feasible space in the direction of the optimal solution. Exactly how the interior-point method generates this search

Received 21 September 1999; revision received 29 March 2000; accepted for publication 19 April 2000.

*Professor Chairman, Department of Mechanical Engineering. Associate Fellow AIAA.
†Research Engineer.

direction determines what kind of interior-point method it is. The successful interior-point methods found in the literature are all immune to degeneracy. This immunity is important in terms of the use of interior-point methods in the context of nonlinear programming problems. Often, the linearized subproblems generated in an SLP strategy are increasingly degenerate if the solution is not critically constrained (i.e., the number of binding constraints is less than the number of design variables). This lack of constraints can occur when the constraints are very nonlinear.[7] The primal-dual path-following interior-point method developed by Vanderbei[8] is used in this work.

III. Primal-Dual Path-Following Interior-Point Algorithm

A. Primal-Dual Problems

Consider the following SLP problem:

$$\begin{aligned} &\text{minimize} && \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to} && A\boldsymbol{x} - \boldsymbol{w} = \boldsymbol{b}, \qquad \boldsymbol{x}, \boldsymbol{w} \geq 0 \end{aligned} \tag{1}$$

where A is an $m \times n$ matrix; $\boldsymbol{c}, \boldsymbol{x}$ are vectors of length n; and $\boldsymbol{w}, \boldsymbol{b}$ are vectors of length m. The variable $\boldsymbol{w}$ is a slack variable such that the constraints could alternatively be written $A\boldsymbol{x} \geq \boldsymbol{b}$. This problem has an associated dual:

$$\begin{aligned} &\text{maximize} && \boldsymbol{b}^T\boldsymbol{y} \\ &\text{subject to} && A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \qquad \boldsymbol{y}, \boldsymbol{z} \geq 0 \end{aligned} \tag{2}$$

where $\boldsymbol{y}$ is a vector of length m and $\boldsymbol{z}$ is a vector of length n. A solution is considered primal feasible if it satisfies the constraints of Eq. (1). Similarly, a solution is considered dual feasible if it satisfies the constraints of Eq. (2). It is known from duality theory that the dual solution serves as a lower bound to the primal solution for all primal-dual feasible points. More important, the primal and dual solutions are equal to the optimum solution of Eq. (1), if one exists. A primal-dual method is one that solves the primal and dual problems simultaneously. Optimality conditions for Eqs. (1) and (2) can be found by satisfying the Karush–Kuhn–Tucker (KKT) conditions[5]:

$$\begin{aligned} &A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \qquad A\boldsymbol{x} - \boldsymbol{w} = \boldsymbol{b}, \qquad WY\boldsymbol{e} = 0 \\ &XZ\boldsymbol{e} = 0, \qquad \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w} \geq 0 \end{aligned} \tag{3}$$

where W, Y, X, and Z are diagonal matrices formed by placing the elements of the corresponding vectors along the diagonal and $\boldsymbol{e}$ is a unit vector of appropriate length. The KKT conditions correspond to critical points of the corresponding Lagrangian function.

B. Primal-Dual Methods and Newton Steps

Since the third and fourth conditions of Eq. (3) are nonlinear, some form of iterative technique must be employed to find the solution of Eq. (3). Newton's method can be used for this purpose. The optimality conditions can be restated as a mapping F from $\Re^{2n+2m}$ to $\Re^{2n+2m}$:

$$F(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}, \boldsymbol{w}) = \begin{bmatrix} A^T\boldsymbol{y} + \boldsymbol{z} - \boldsymbol{c} \\ A\boldsymbol{x} - \boldsymbol{w} - \boldsymbol{b} \\ WY\boldsymbol{e} \\ XZ\boldsymbol{e} \end{bmatrix} = 0, \qquad \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w} > 0 \tag{4}$$

Starting with a first order Taylor's series approximation of the terms of Eq. (4) around the current solution and ignoring higher-order terms, one obtains a search direction by employing the Jacobian $J(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y}, \boldsymbol{w})$ such that

$$J(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w}) \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{w} \\ \Delta\boldsymbol{z} \end{bmatrix} = -F(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w}) \tag{5}$$

which can be rewritten as

$$\begin{bmatrix} 0 & A^T & 0 & I \\ A & 0 & -I & 0 \\ 0 & W & Y & 0 \\ Z & 0 & 0 & X \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{w} \\ \Delta\boldsymbol{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} - A^T\boldsymbol{y} - \boldsymbol{z} \\ \boldsymbol{b} - A\boldsymbol{x} + \boldsymbol{w} \\ -WY\boldsymbol{e} \\ -XZ\boldsymbol{e} \end{bmatrix} \tag{6}$$

Note that, if the current point is strictly feasible, the first two terms of the right-hand side of equation Eq. (6) will be zero. This feasibility shows that such a method does not require a feasible starting point but can progress toward feasibility and optimality at the same time. In fact, if the problem is well-posed (i.e., it is not unbounded and has a feasible solution), the algorithm should proceed monotonically toward both primal and dual feasibility (at least until machine precision interfaces).[8] This fact can be used to identify such problems as primal unboundedness (if the dual infeasibility increases) or primal infeasibility (if the primal infeasibility increases).

It is to be noted that taking the full step obtained by the solution of Eq. (6) may violate the nonnegativity restrictions in Eq. (4) that often result in divergence of the algorithm. For this reason, the next point is obtained by scaling the step length by a factor $0 < \alpha \leq 1$ such that

$$(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w})_{\text{new}} = (\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w})_{\text{old}} + \alpha(\Delta\boldsymbol{x}, \Delta\boldsymbol{z}, \Delta\boldsymbol{y}, \Delta\boldsymbol{w}) \tag{7}$$

where α is determined by a heuristic that ensures strict nonnegativity for all variables. If $\alpha = 1$ in any given iteration, feasibility will be achieved in that step and all subsequent iterations will remain feasible. Since the primal and dual solutions may not approach the constraints at the same speed, it is possible that a different step length correction α could be taken for the primal and dual problems.

The method described above is known as the *primal-dual affine scaling* algorithm. This algorithm attempts to move toward the optimal solution (and perhaps the feasible space, if necessary) of the primal and dual problems at each Newton step. When the current solution is far from any constraint boundaries, this movement results in rapid progress toward the optimum. However, as the solution approaches the constraints, the Newton direction may intersect with the constraint boundaries. This intersection results in a step length α that is much less than 1 in order to satisfy the nonnegativity conditions. In practice, the primal-dual affine scaling algorithm is found to perform poorly.

C. Central Path and Interior-Point Algorithms

The central path is a continuous curve of strictly feasible solutions that avoids points near the constraint boundaries. The curve is parameterized by a variable μ and solves the following system, which can be derived by rewriting the nonnegativity constraints as logarithmic penalties and finding the critical points of the corresponding Lagrangian[5]:

$$\begin{aligned} &A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \qquad A\boldsymbol{x} - \boldsymbol{w} = \boldsymbol{b}, \qquad WY\boldsymbol{e} = \mu\boldsymbol{e} \\ &XZ\boldsymbol{e} = \mu\boldsymbol{e}, \qquad \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}, \boldsymbol{w} > 0 \end{aligned} \tag{8}$$

Notice that Eq. (8) differs from the KKT conditions in Eq. (3) by the addition of μ to the right-hand side of the third and fourth expressions, as well as the strict positivity condition imposed on $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}$, and $\boldsymbol{w}$. It can easily be seen that in the limit, as μ goes to zero, the central path converges to the KKT condition, and, hence, to the optimal solution of the original linear problem Eq. (1). An algorithm that generates a series of points on or near the central path while reducing the value of μ (i.e., a path-following algorithm) will converge to the optimal solution.[9]

Also, the tendency to follow a path away from the constraint boundaries or into the interior of the feasible space yields the origin of the name, interior-point method. This is quite different from the simplex method, which can be interpreted geometrically as an algorithm that moves along the boundaries from one constraint intersection to another attempting to locate the minimum intersection (which movement will always yield the minimum to a nondegenerate LP problem).

Following a similar progression as in Sec. III.B, the central path found in Eq. (8) results in the following system:

$$\begin{bmatrix} 0 & A^T & 0 & I \\ A & 0 & -I & 0 \\ 0 & W & Y & 0 \\ Z & 0 & 0 & X \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{y} \\ \Delta \boldsymbol{w} \\ \Delta z \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} - A^T \boldsymbol{y} - z \\ \boldsymbol{b} - A\boldsymbol{x} + \boldsymbol{w} \\ \mu \boldsymbol{e} - WY\boldsymbol{e} \\ \mu \dot{\boldsymbol{e}} - XZ\boldsymbol{e} \end{bmatrix} \tag{9}$$

that has a solution for each $\mu > 0$. The solution to this system does not move directly toward the optimum of the original problem but toward some point on the central path (and toward feasibility, if necessary). A path-following algorithm uses a modified Newton step to solve Eq. (9), while simultaneously reducing the value of μ in some prescribed fashion. Although it is unlikely that the solution to Eq. (9) will reach a point exactly on the central path in one iteration, it is not necessary to do so. Both theory and experience[9] show that a neighborhood around the central path also yields good results. Following the central path has the advantage of avoiding conflicts with constraint boundaries, but staying in a small neighborhood of the central path results in what is called *short-step interior-point* methods that do not necessarily progress rapidly toward the optimum.

D. Implementation of Linear Algebra

Two important aspects are to be considered in implementing the linear algebra relations in interior-point algorithms. The first aspect is related to sparsity. Since the advantages of interior-point algorithms over other linear solvers (like simplex) are best shown on large problems, it is likely that sparse linear algebra techniques will become important in the implementation of these algorithms.[6] For large problems, the constraint matrix A in Eq. (1) is very sparse (typically less than 10% nonzero elements). The storage of a large number of zeros is both useless and costly. Even today when computer memory is "cheap," it is not difficult to construct a problem that exceeds the physical RAM of most workstations. Also, it is terribly inefficient to grind through thousands of floating point operations when the data involved are mostly zero. It is much better to only use the nonzero elements of the sparse matrix. Therefore, suitable sparse linear algebra utilities are to be used to solve the problem efficiently.

In relation to matrix sparsity, one must consider the effects of factorization on storage requirements. If one naively applies a factorization such as Cholesky's on a sparse system of equations, the result may no longer be sparse. If factorization is viewed in terms of a Gaussian elimination, the process of factorization amounts to a series of row reductions until a desired form is found. At a given stage of the Gaussian elimination, it is desirable to "zero out" elements below the diagonal. If the current row/column is dense (i.e., has many nonzero terms) these terms will be added to or subtracted from all of the rows/columns that have not yet been reduced. In the worst case, if the current row is completely dense, the resulting factorization will be completely dense from this point on. This is known as *catastrophic fill-in*. To combat this problem, it is possible to permute the rows and columns of the matrix to minimize the amount of fill-in. Many fill-in reducing heuristics are available, with the minimum-degree ordering and minimum local fill-in heuristics being the most popular.[6]

The second aspect involves the form of the equations solved. Typically, equations such as Eq. (9) are not solved "as is." They are first reduced until just the Δx and Δy components are left, so that

$$\Delta \boldsymbol{w} = Y^{-1}(\mu \boldsymbol{e} - WY\boldsymbol{e} - W\Delta \boldsymbol{y})$$

$$\Delta z = X^{-1}(\mu \boldsymbol{e} - XZ\boldsymbol{e} - Z\Delta \boldsymbol{x}) \tag{10}$$

$$\begin{bmatrix} -X^{-1}Z & A^T \\ A & Y^{-1}W \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} - A^T \boldsymbol{y} - \mu X^{-1}\boldsymbol{e} \\ \boldsymbol{b} - A\boldsymbol{x} + \mu Y^{-1}\boldsymbol{e} \end{bmatrix} \tag{11}$$

The system in Eq. (11) is known as the *augmented equations* or the *reduced KKT equations*. An inspection of Eq. (11) shows that the matrix is symmetric but not positive definite. Although some researchers have used the Bunch–Parlett factorization for symmetric, indefinite, systems.[6] Vanderbei has shown that this form (which he calls *symmetric quasi-definite*) can be factored with a LDL^T version of Cholesky factorization that is much less costly than the Bunch–Parlett factorization.[10] The latter method is used in the present implementation.

E. Extensions

The simple problem in Eq. (1) can easily be modified to include such things as upper and lower bounds on the variable $\boldsymbol{x}$. The linear problem is posed in the form of

$$\begin{aligned} \text{minimize} \quad & f + \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} - \boldsymbol{w} = \boldsymbol{b} \\ & \boldsymbol{x} - g = l \\ & \boldsymbol{x} + t = u \\ & g, \boldsymbol{w}, t \geq 0 \\ & \boldsymbol{x} \text{ free} \end{aligned} \tag{12}$$

which differs from Eq. (1) in the addition of a scalar f, upper (u) and lower (l) bounds on $\boldsymbol{x}$, and two new slack variables g and t. This form differs from the standard form required by many linear solvers in that $\boldsymbol{x}$ no longer has a nonnegativity restriction. This lack of a restriction is allowed by the addition of the lower bound l. Although any problem can easily be converted into a nonnegative problem, in the case of a free variable (one with no finite upper or lower bound), this conversion requires splitting the variable into two nonnegative pieces whose sum is the desired value. This split has been known to cause numerical instabilities in some problems.[8] The form suggested here does not have this problem. Furthermore, the absence of this problem is desirable for engineering implementation where $\boldsymbol{x}$ may be required to take on negative values. The dual of the problem is given by

$$\begin{aligned} \text{maximize} \quad & f + \boldsymbol{b}^T \boldsymbol{y} + l^T z - u^T s \\ \text{subject to} \quad & A^T \boldsymbol{y} + z - s = c \\ & \boldsymbol{y} - \boldsymbol{v} = 0 \\ & z, v, s \geq 0 \\ & \boldsymbol{y} \text{ free} \end{aligned} \tag{13}$$

where v is a new slack variable and $\boldsymbol{y}$ has no restriction.

Finally, it is necessary to define the measures used to determine the convergence of the algorithm. Two measures will be used to measure the primal infeasibility and dual infeasibility, respectively. A third measure will be used to determine the number of significant figures of agreement between the primal and dual objective values. The number of figures of agreement can be used as a termination criteria because the primal and dual solutions are the same at the optimum. These measures can be defined as

$$\text{primal infeasibility} = \frac{\sqrt{\|\boldsymbol{\rho}\|^2 + \|\tau\|^2 + \|\upsilon\|^2}}{\|\boldsymbol{b}\| + 1} \tag{14}$$

$$\text{dual infeasibility} = \frac{\sqrt{\|\boldsymbol{\sigma}\|^2 + \|\beta\|^2}}{\|\boldsymbol{c}\| + 1} \tag{15}$$

significant figures

$$= \max\left(-\log \frac{|\text{primal objective} - \text{dual objective}|}{|\text{primal objective}| + 1}, 0 \right) \tag{16}$$

where the primal and dual objectives are given in Eqs. (12) and (13), respectively.

IV. Sequential Linear Programming

A. Standard Method

In SLP, a general nonlinear programming problem is solved by sequentially solving a series of LP problems. Kelly[11] first introduced the idea under the name of a "cutting-plane" algorithm. If a problem is convex, the linearized constraints will generate a polygonal approximation to the nonlinear constraints by successively adding

new linear constraints after solving each linear problem. The resulting linear optimum will always lie outside the feasible space of the nonlinear problem. By continuing the linearization process, the optimum value can be found to within a given tolerance.[12,13] If the problem is not convex, the problem becomes much more difficult because some heuristic must be employed to determine when one or more of the added constraints should be deactivated (so that they do not block feasible space). An aspect of this formulation is that the problem size grows every time new linearized constraints are added.

An alternate approach to SLP that does not grow in size is to solve for a box constrained step direction using a linearized approximation of the objective and constraints.[14] Given a nonlinear programming problem

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ \text{subject to} \quad & g_j(\boldsymbol{x}) \le 0, && j = 1, mi \\ & h_k(\boldsymbol{x}) = 0, && k = 1, me \\ & lb \le \boldsymbol{x} \le ub \end{aligned} \tag{17}$$

The first-order Taylor's series expansion about some starting point x^0 results in the approximate problem

$$\begin{aligned} &\text{minimize} \quad f(x) \approx f(x^0) + \nabla f(x^0) \cdot \delta x \quad \text{subject to} \\ &g_j(x) \approx g_j(x^0) + \nabla g_j(x^0) \cdot \delta x \le 0, \qquad j = 1, mi \\ &h_h(x) \approx h_k(x^0) + \nabla h_k(x^0) \cdot \delta x \le 0, \qquad k = 1, me \\ &l \le \delta x \le u \end{aligned} \tag{18}$$

where l and u are lower and upper bounds imposed on the step direction δx to avoid the possibility of an unbounded problem. Once Eq. (18) is solved, the new solution can be obtained as

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \delta \boldsymbol{x} \tag{19}$$

The SLP algorithm used in the present work is an efficient SLP algorithm that uses fuzzy heuristics to adaptively modify algorithm parameters on the fly. The traditional SLP algorithm evaluates the solution of each LP to decide if the new information should be accepted and used to move to a new candidate design or rejected (after which some parameters are adjusted and another LP is solved[14]). This results in an algorithm that may spend many computational cycles working on the solution to a linear problem that will only be discarded. The authors developed a SLP algorithm that uses fuzzy heuristics to adaptively modify the algorithm behavior on-the-fly so that no intermediate LP solutions need be discarded.[15] Furthermore, the fuzzy heuristics are used to adaptively modify constraint boundaries to reduce the tendency of the SLP algorithm to produce infeasible intermediate designs.

B. Fuzzy Heuristics

The SLP algorithm used in this work differs from the conventional algorithm by the addition of fuzzy heuristics, the addition of boundary control factors, and the elimination of the conditional checking to see if the current solution is less infeasible than the previous one. The significance of these changes is that no solution is ever "discarded." All optimization calculations are considered valuable, and there are no wasted computational cycles. Furthermore, the inclusion of the boundary control factor in setting up the linear problem helps modify the constraints to bring the solution back to the feasible space instead of just relying on a reduction in the move limits to do so. More specifically, the following heuristics are used:

1) If the design point is feasible, then increase the move limits slightly.

2) If the design point is very infeasible, then slightly reduce move limits and tighten constraints.

3) If the design point is near binding constraints, then rapidly reduce the move limits and tighten constraints slightly.

4) If the search direction is unchanged, then slightly increase the move limits.

5) If the search direction has reversed, then rapidly decrease the move limits.

Fuzzy logic was used to interpret the descriptive and qualitative terms such as "slightly," "very infeasible," "rapidly reduce," and "near binding constraint." The first heuristic is an attempt to speed movement when the design point is inside the feasible space and constraint boundaries are far away. The second heuristic attempt to encourage movement toward feasible space by tightening the constraints aggressively to shift the next solution closer to the feasible space. The third heuristic is aimed at rapidly converging, once binding constraints are identified, by quickly shrinking the move limits. The boundary control factor pulls the constraints back slightly to stay closer to feasible space. The fourth and fifth heuristics modify the move limits according to changes or trends in the search direction. This modification prevents premature termination caused by a series of iterates that must "slide along" a constraint before encountering a fully constrained condition.

V. Numerical Results

The numerical efficiency of the interior-point method used in an SLP strategy is demonstrated by considering three numerical examples. The first problem, a part stamping problem, is used to demonstrate the advantage of the interior-point method as the problem size increases. The part stamping problem also shows the effects of degeneracy on the algorithm performance. The second and third problems, chosen to be more realistic engineering problems, are taken from the Constrained and Unconstrained Testing Environment (CUTE) test suite.[16] The second example, called ROTDISK, considers the design of a jet engine rotor for minimal weight under constraints of stress and profile. The third example, called SREADIN3, is a nonlinear optimal control problem arising from tidal modeling.

Sequential linear programming was used to solve each sample problem with three different linear solvers. The first solver was based on the primal-dual path-following interior-point method. The second and third linear solvers were primal and dual simplex LP solvers using sparse matrix storage from the commercial package CPLEX (Ref. 17).

A. Part Stamping Problem

The parameters used to define a part stamping problem are shown in Fig. 1. The objective is to minimize the total area ab used to stamp out n-disk number of parts without allowing overlap. This problem is stated mathematically as

$$\begin{aligned} \text{Find } \chi = [&x_1 \quad x_2 \quad \cdots \quad x_{n\text{-disk}} \quad y_1 \quad y_2 \quad \cdots \quad y_{n\text{-disk}} \quad a \quad b]^T \\ &\text{to minimize} && f(\chi) = ab \\ &\text{subject to} \\ & x_i + R_i \le a, && i = 1, n\text{-disk} \\ & y_i + R_i \le b, && i = 1, n\text{-disk} \\ & (x_i - x_j)^2 + (y_i - y_j)^2 \ge (R_i + R_j)^2, && i = 1, n\text{-disk} - 1 \\ & && j = i + 1, n\text{-disk} \\ & x_i \ge R_i, && i = 1, n\text{-disk} \\ & y_i \ge R_i, && i = 1, n\text{-disk} \end{aligned} \tag{20}$$

Notice that both the objective and the third set of constraints (preventing overlap of adjacent parts), are nonlinear. The size of

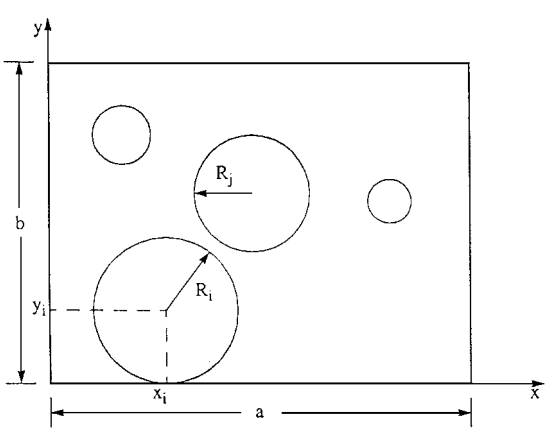

Fig. 1 Parameters defining the part stamping problem.

this problem grows very rapidly with the number of parts (n-disk). The number of variables n and number of constraints (not counting side constriants) m are given by (Table 1)

$$n = 2(n\text{-disk}) + 2, \qquad m = 2(n\text{-disk}) + \sum_{i=1}^{(n\text{-disk})-1} i \tag{21}$$

The part stamping problem was solved for various number of parts with a constant radius of 1. A random starting point was chosen for each case in which the parts were scattered in an intentionally loose arrangement. Algorithm specific data for each problem were set as

initial move limit, $\Delta X = n$-disk
termination criteria (% of initial move limit), $\varepsilon = 0.01$
radii of all parts, $R_i = 1$

Optimization results for the part stamping problem with various number of stamped parts are shown in Table 2. Once an algorithm stops, the result (relative minimum) was also verified through a visual inspection of the final packing arrangement. MATLAB® was used to plot the arrangement of the packed disks. The results are tabulated for each of the three LP solvers in terms of the final optimal objective obtained [$f(X^*)$], CPU time, and the number of LP subproblems solved.

The primal simplex-based SLP either failed or converged to an erroneous solution in all cases. This phenomenon is most likely due to the structure of the problem. Since the number of constraints is much greater than the number of variables ($m \gg n$), the primal problem tends to be quite ill-posed. On the other hand, this problem structure should favor the dual simplex method, which, in fact, performed much better. The primal simplex method finished in the smallest CPU time in nearly every case and provided adequate results for the small- and medium-sized problems (n-disk ≤ 50). However, it failed to give good results for larger problems as discussed next.

The most important result is that the interior-point-based SLP terminated with the lowest objective value in all but one case and produced significantly better results for larger problems (n-disk ≥ 100). The simplex-based LP solvers (which solve for basic or vertex solutions) seem to prematurely converge to local minima much faster than the interior-point-based LP solver. This happens because the simplex-based LP algorithms solve for basic (or vertex) solutions that are susceptible to degeneracy. Meanwhile, the interior-point-based algorithm approaches the optimum from within the interior of the feasible space and is immune to degeneracy. Although the total solution time for the interior-point-based solver was longer, the increase in time was because more LP subproblems had to be solved in order to produce a much better solution to the original nonlinear problem.

Table 1 Size of the part stamping problem

Number of parts n-disk	Number of variables n	Number of constraints m
5	12	20
10	22	65
20	42	230
30	62	495
50	102	1,325
100	202	5,150
150	302	11,475
200	402	20,300

B. Turbine Rotor Design

The turbine rotor design problem is concerned with the design of a minimal weight turbine disc subject to constraints on thermal stresses, stresses due to external loads, stresses due to rotation, and certain geometric constraints.[18] The turbine rotor design problem is formulated with the following parameters:

$E(r)$	=	modulus of elasticity
R_i	=	inner radius
R_o	=	outer radius
$t(r)$	=	temperature as a function of radial distance
$W(r)$	=	thickness as a function of radial distance
$W_M(r)$	=	given maximum thickness function
$W_m(r)$	=	given minimum thickness function
$\alpha(r)$	=	coefficient of thermal expansion
μ	=	specific mass of the metal
ν	=	the Poisson's ratio
σ_r	=	radial stress, a function of radial distance
σ_{ro}	=	outer radial stress (due to the centrifugal force of the blades)
$\sigma_{rM}(t, r)$	=	given maximum radial stress function
σ_t	=	circumferential stress, a function of radial distance
$\sigma_{tM}(t, r)$	=	given maximum circumferential stress function
σ_w	=	axial stress
ω	=	angular speed

The model is defined by the following governing equations:

$$r\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\sigma_t(r) - \sigma_r(r)}{E(r)} + \alpha(r)\cdot t(r)\right] = -(1+\nu)\frac{\sigma_t(r) - \sigma_r(r)}{E(r)}$$

$$\frac{\mathrm{d}}{\mathrm{d}r}[W(r)\cdot r\cdot \sigma_r(r)] = W(r)\left[\sigma_t(r) - \mu\omega^2 r^2\right]$$

$$\sigma_r(R_o) = \sigma_{\text{ro}}, \qquad \sigma_t(R_i) = 0 \tag{22}$$

The optimization problem is formulated as

Find $\chi = W(r)$ (at discrete locations) to minimize $W(r)$ subject to

$$\operatorname{Max}_r\{\sigma_r(r) - \sigma_{rM}[t(r), r]\} \leq 0$$

$$\operatorname{Max}_r\{\sigma_t(r) - \sigma_{tM}[t(r), r]\} \leq 0$$

$$\operatorname{Max}_r\{\sigma_r(r) - \mathrm{d}\sigma_t(r)\} \leq 0 \qquad (\mathrm{d} \text{ is a given parameter})$$

$$\int_{R_i}^{R_o} \sigma_r(r)W(r)\,\mathrm{d}r - J\left[\frac{1}{R_o - R_i}\int_{R_i}^{R_o} W(r)t(r)\,\mathrm{d}r\right] \leq 0$$

(J is a given function)

$$\operatorname{Max}_r\{W(r) - W_M(r)\} \leq 0, \qquad \operatorname{Max}_r\{W_m(r) - W(r)\} \leq 0 \tag{23}$$

Table 2 Summary of results for part stamping problems

Number of parts	Interior-point $f(X^*)$	Interior-point CPU time, h:m:s.ms	Interior-point LPS solved	Primal simplex $f(X^*)$	Primal simplex CPU time, h:m:s.ms	Primal simplex LPs solved	Dual simplex $f(X^*)$	Dual simplex CPU time, h:m:s.ms	Dual simplex LPs solved
5	22.928	0.38	11	34.853	0.48	31	23.287	0.36	27
10	43.932	1.10	13	75.714	0.86	31	44.937	0.72	29
20	88.777	4.40	13	135.86	4.80	61	87.197	2.08	31
30	126.08	30.32	37	4,973.3[a]	2.64	8	155.96	4.96	28
50	201.23	1:27.87	30	28,586	10.86	23	203.00	33.37	32
100	398.08	11:51.94	32	47,093	3:11.59	30	1,018.2	2:56.89	30
150	600.11	44:50.11	32	144,080	12:23.74	24	896.52	16:63.61	29
200	781.10	2:01:37.02	32	801,767[b]	14:41.97	4	1,367.3	57:02.88	29

[a] Could not optimize linear subproblem. [b] Terminated with error.

Table 3 Summary of results for turbine rotor design and optimal control problem

Problem	$f(X^*)$	Interior-point CPU time, h:m:s.ms	LPs solved	Primal simplex CPU time, h:m:s.ms	LPs solved	Dual simplex CPU time, h:m:s.ms	LPs solved
Turbine rotor design (ROTDISK)	7.872	4:18.37	14	4:56.21	14	5:18.99	14
Optimal control (SREADIN3)	−0.1526	21:55.94	16	34:07.86	26	35:31.07	26

The thickness function $W(r)$ was discretized into 180 steps resulting in 905 variables (180 free variables, 1 variable bounded only from above, 180 variables bounded only from below, 531 bounded from above and below, and 13 fixed variables) and 1081 constraints (360 linear equalities, 361 linear inequalities, and 360 nonlinear equalities).

The numerical results for the rotor design problem are shown in Table 3. For this problem, all three LP solvers were able to solve the rotor design problem with 14 linear programming problems. Each algorithm also yielded the same optimal result. However, the interior-point method found the solution 15% faster than the primal simplex method and 23% faster than the dual simplex method.

C. Optimal Control Problem

The optimal control problem involves the design of a turbine for power generation based on tidal movements.

The following problem formulation comes from Lyle and Nichols.[19] The parameter definitions are as listed:

$\tilde{A}$ = basin surface area
$f(t)$ = tidal level
$h(t)$ = head difference, defined as $h(t) \equiv \eta(t) - f(t)$
$Q_1(h)$ = maximum sluicing capacity of the turbines for head h
$Q_2(h)$ = maximum flow for turbine use for head h
T = tidal period
$\eta(t)$ = basin level above some datum point

The optimal control problem can be formulated as

$$\max E = \int_0^T e[X(h)u(t), h(t)]\,\mathrm{d}t$$

$$\text{subject to} \quad \eta(t) = -\frac{1}{\tilde{A}} X(h)u(t)$$

$$\eta(0) = \eta(T), \qquad u(t) \in [0, 1] \tag{24}$$

where

$$X(h) \equiv \begin{cases} -Q_1(h), & h \le 0 \\ Q_2(h), & h > 0 \end{cases} \tag{25}$$

$$e[X(h)u(t), h(t)] \equiv \begin{cases} -Q_1 uh, & h \le 0 \\ Q_2 uh, & h > 0 \end{cases} \tag{26}$$

The problem formulation for SREADIN3 in the CUTE test suite also includes special variable scaling.[16] The optimal control problem was formulated with 2002 variables (bounded from above and below) and 1001 constraints (one linear equality and 1000 nonlinear equalities).

Results for the optimal control problem are also summarized in Table 3. Once again, all three algorithms yielded basically the same optimal objective. However, the interior-point method did so in just 16 LPs whereas the simplex methods required 26 LPs to converge. The difference is possibly explained by the fact that the interior-point method yields nonbasic solutions. On the one hand, if the solution is degenerate near the optimum (as it could be with 1000 nonlinear constraints), the simplex methods would give basic (or vertex) solutions and thus require a very small bounding box to converge to the required tolerance. On the other hand, the interior-point method would produce nonbasic solutions that would be closer to the degenerate optimal point. Whatever the cause, the interior-point method solved the optimal control problem 56% faster than the primal simplex method and 62% faster than the dual simplex method.

VI. Conclusions

The efficient solution to three large-scale design optimization problems with an interior-point-based SLP algorithm was demonstrated. Linear problems with hundreds or thousands of design variables and constraints can be efficiently solved with an adaptive SLP algorithm using a primal-dual path-following interior-point LP solver. The interior-point based SLP algorithm appears to have superior performance for many classes of problems. For the three problems solved in this study, the interior-point based SLP algorithm consistently performed both faster and more reliably than either primal or dual simplex-based SLP.

References

[1]Karmarkar, N., "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, Vol. 8, No. 4, 1984, pp. 373–395.

[2]Hooker, J. N., "Karmarkar's Linear Programming Algorithm," *Interfaces*, Vol. 16, No. 4, 1986, pp. 75–90.

[3]Vanderbei, R. J., "Interior-Point Methods: Algorithms and Formulations," *ORSA Journal on Computing*, Vol. 6, No. 1, 1994, pp. 32–34.

[4]Anstreicher, K. M., "Potential Reduction Algorithms," *Interior Point Methods in Mathematical Programming*, edited by T. Terlaky, Kluwer, Dordrecht, The Netherlands, 1996, Chap. 4, pp. 125–158.

[5]Wright, S. J., *Primal-Dual Interior Point Methods*, Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[6]Anderson, E. D., Gondzio, J., Meszaros, C., and Xu, X. J., "Implementation of Interior Point Methods for Large Scale Linear Programming," Logilab, HEC Geneva Section of Management Studies, TR 1996.3, Univ. of Geneva, Geneva, 1996.

[7]Vandengerghe, L., and Boyd, S., "Semidefinite Programming," *SIAM Review*, Vol. 38, No. 1, 1996, pp. 49–95.

[8]Vanderbei, R. J., "LOQO: An Interior Point Code for Quadratic Programming," Program in Statistics and Operations Research, TR SOR 94-15, Princeton Univ., Princeton, NJ, 1995.

[9]Gonzaga, C. C., "Path-Following Methods for Linear Programming," *SIAM Review*, Vol. 34, No. 2, 1992, pp. 167–224.

[10]Vanderbei, R. J., "LOQO User's Manua: Version 2.21," Program in Statistics and Operations Research, TR SOR 96-07, Princeton Univ., Princeton, NJ, 1995.

[11]Kelly, J. E., "The Cutting Plane Method for Solving Convex Programs," *Journal of SIAM*, Vol. 8, No. 4, 1960, pp. 703–712.

[12]Bertsekas, D. P., *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1995.

[13]Rao, S. S., *Engineering Optimization: Theory and Practice*, 3rd ed., Wiley, New York, 1996, Chap. 7.

[14]Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw–Hill, New York, 1984, Chap. 6.

[15]Mulkay, E. L., and Rao, S. S., "Fuzzy Heuristics for Sequential Linear Programming," *Journal of Mechanical Design*, Vol. 120, No. 1, 1998, pp. 17–35.

[16]Bongartz, I., Conn, A. R., Gould, N., and Toint, P. L., "CUTE: Constrained and Unconstrained Testing Environment," *ACM Transactions on Mathematical Software*, Vol. 21, No. 1, 1995, pp. 123–160.

[17]CPLEX, *Using the CPLEX Callable Library, Version 4.0*, CPLEX Optimization, Inc., Incline Village, NV, 1995.

[18]Nigoghossian, J. P., "The Optimization of Jet Engine Discs," *Optimization and Design*, edited by M. Avriel, M. J. Rijckaert, and D. J. Wilde, Prentice–Hall, Englewood Cliffs, NJ, 1973, pp. 346–349.

[19]Lyle, S., and Nichols, N. K., "Numerical Methods for Optimal Control Problems with State Constraints," Mathematics Dept., Univ. of Reading, TR 8/91, Reading, England, U.K., 1991.

A. Chattopadhyay
Associate Editor