ANALYSIS OF A NONHIERARCHICAL DECOMPOSITION ALGORITHM

by

Jayashree Shankar

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science and Applications

APPROVED:

Layne T. Watson
Layne T. Watson

Raphael T. Haftka

Calvin J. Ribbens
Calvin J. Ribbens

August, 1992

Blacksburg, Virginia

ANALYSIS OF A NONHIERARCHICAL DECOMPOSITION ALGORITHM FOR LARGE SCALE OPTIMIZATION

by

Jayashree Shankar

Committee Chairman: Layne T. Watson

Computer Science

(ABSTRACT)

Large scale optimization problems are tractable only if they are somehow decomposed. Hierarchical decompositions are inappropriate for some types of problems and do not parallelize well. Sobieszczanski-Sobieski has proposed a nonhierarchical decomposition strategy for nonlinear constrained optimization that is naturally parallel. Despite some successes on engineering problems, the algorithm as originally proposed fails on simple two dimensional quadratic programs.

Here, the algorithm is carefully analyzed by testing it on simple quadratic programs, thereby recognizing the problems with the algorithm. Different modifications are made to improve its robustness and the best version is tested on a larger dimensional example. Some of the changes made are very fundamental, affecting the updating of the various tuning parameters present in the original algorithm.

The algorithm involves solving a given problem by dividing it into subproblems and a final coordination phase. The results indicate good success with small problems. On testing it with a larger dimensional example, it was discovered that there is a basic flaw in the coordination phase which needs to be rectified.

ACKNOWLEDGEMENTS.

At the outset I would like to thank Dr. Layne T. Watson for taking me in as his student and guiding me through my Masters program. His willingness to share his knowledge is quite unmatched. His wisdom and experience of technically sound publication has helped me immensely in the preparation of this thesis.

I am very grateful to Dr. Raphael T. Haftka for serving on my committee and for being a source of constant encouragement. I also acknowledge his experience and knowledge which were instrumental in guiding us in our research.

My thanks are also due to Dr. Calvin J. Ribbens for serving on my committee and for all his suggestions regarding my research and thesis.

I would like to thank my husband Shankar for being by my side and helping me during the rough times.

Last but not the least I thank my daughter, little Nikhila, for being so patient and understanding and letting "mommy" work those long and late hours.

It is my greatest pleasure to dedicate this thesis to my loving parents whose dream it was to see me accomplish this and whose love helped me fulfill it.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION.

Many engineering problems involve large scale optimization over many different disciplines. As is the case with many large scale problems, a decomposition of the problem into subproblems helps reduce the time and complexity of solution. The strategy governing the decomposition of a large scale problem can directly affect the ease and accuracy of the solution. The concept of a linear decomposition strategy by Sobieski [33] has been used with good results in a number of cases. This method works very well in the case of a system that is amenable to such a decomposition, i.e., when subsystems can be laid out clearly in a hierarchical fashion.

For a system with many interdependencies between the probable subproblems, using a linear decomposition strategy implies choosing one subsystem before another, thereby establishing an artificial hierarchy. The order chosen will affect the solution iterates, making this strategy illsuited or even nonconvergent for such nonhierarchic problems.

These considerations led Sobieski [32] to propose a new nonhierarchic decomposition strategy. The algorithm involves introducing various coefficients to represent the interdependencies (responsibility and trade-off coefficients) between the subsystems and solving the subsystems concurrently. The coordination phase for these subproblems involves updating these coefficients based on sensitivity information. This algorithm has been tested with success on engineering problems by Bloebaum/Hajela/Sobieski [7]. Also, heuristics [6] are employed to help with the solution of each subsystem, where human intervention is possible.

In this thesis we carry out a detailed analysis of this algorithm by Sobieski [32] and its feasibility. Since nonlinear optimization can be reduced to a series of quadratic programs, it is appropriate to study this new algorithm first on quadratic programs. Thus, this thesis first studies the various tuning parameters occurring in this algorithm, using a model quadratic programming problem. A series of experiments shows that modifications to the algorithm as originally proposed by Sobieski [32] are necessary for convergence. This modified algorithm is then used to solve problems involving a number of subsystems, each with a varying number of design variables.

The tests are carried out on quadratic programming (QP) problems of different dimensions. The decomposition then yields subproblems which are also QP problems. The method employed to solve these smaller QP problems is an active set strategy described in Fletcher [12]. Also optimization packages such as MINOS [23] and QPSOL [16] were used to verify the answers.

QPSOL is an optimization package that solves quadratic and linear programming problems. The coordination phase of the algorithm presented in this thesis is a linear programming problem that is solved in order to update the coefficients that mark the coupling of the subsystems. LPSOL (part of QPSOL) is used to solve this linear programming problem.

An alternative to the coordination phase by Renaud/Gabriele is given in [27]. Here, instead of having the coordination phase give an updated vector of the interdependency coefficients, an approximation to the global problem is solved to get an updated design vector. In [37] Sobieski gives two alternative ways to solve the coordination optimization phase as is used in [33].

An algorithm for optimization based design of non-hierarchic systems is given by Wu/Azarm in [48]. It uses decomposition techniques to solve the given problem in two stages. Details for handling multidisciplinary systems are not included. In [24] Pan/Diaz give another method for optimization by non-hierarchic decomposition . Here, the subspace optimizations are solved sequentially and not concurrently and global convergence is guaranteed. This is as a result of the subsystems being solved sequentially and by a method given in [24] for moving away from pseudo optimal points (points that are optimal for every subsystem, but not optimal for the global problem) and towards the optimal solution. An obvious disadvantage is the sequential nature of the subspace optimizations.

The fact that the subsystems are solved concurrently in the algorithm given in this thesis makes it naturally parallel. The sensitivity derivatives can also be obtained in parallel too. Future work could involve parallelization of the best serial algorithm given in this thesis after the coordination optimization phase is modified to solve the problems described in

Chapter 7. Towards that end the papers given in [8], [40], [41], [19], [38], and [20] may be useful. They may help in the efficient parallelization of the algorithm.

Chapter 2 includes a detailed description of the problem and the notation employed, and also a description of the simple decomposition technique that originally introduced the concept of subspace optimization. Chapter 3 explains the algorithm with approximate coupling in detail. The pseudocode for the algorithm is given in Chapter 4. The results of some initial tests of the algorithm as it was proposed by Sobieski in [32] are tabulated in Chapter 5. Chapter 6 describes the modifications that were made to the original algorithm. Chapter 7 records the examples that the algorithm was tested on and the results obtained. The quadratic programming algorithm used to solve the individual subproblems is described in Chapter 8. Also, the algorithms used in the optimization packages QPSOL and MINOS are given there. Chapter 9 gives an account of the historical development of the current version of the algorithm from the original one. Chapter 10 concludes.

2. PROBLEM STATEMENT AND SIMPLE SEQUENTIAL DECOMPOSITION.

2.1 Problem Statement.

Although in this thesis we are only dealing with quadratic programming problems, this algorithm [32] is meant to solve any nonlinear programming problem. Therefore the algorithmic details are for any nonlinear programming problem.

Consider the following nonlinear programming problem (NLP),

$$\begin{aligned} &\min_x \Theta(x) \\ \text{subject to} \quad &g(x,y) \leq 0, \\ &h(x,y) = 0, \end{aligned} \tag{2.1}$$

where $x \in E^n$, $y \in E^p$, g is an m-dimensional vector function and h is a p-dimensional vector function. x is the set of design variables and y is the set of behavior variables which are the unknowns in each subsystem.

Subspace optimization involves solving a problem by solving a set of subproblems. The global problem is divided into subproblems according to some logical way of partitioning the problem. In the scheme that originally introduced subspace optimization, which we shall refer to as simple sequential decomposition, there is not much more to the solution of the global problem than the solution of these subproblems iteratively until the same solution vector solves each of the individual subproblems.

To outline the differences between the current scheme and simple decomposition, we introduce the following terminology:

$$\begin{gathered} x = (X^1, X^2, \ldots, X^N), \qquad X^i \in E^{n_i}, \qquad n_1 + n_2 + \ldots + n_N = n, \\ y = (Y^1, Y^2, \ldots, Y^N), \qquad Y^i \in E^{p_i}, \qquad p_1 + p_2 + \ldots + p_N = p, \\ g = \begin{pmatrix} g^1 \\ \vdots \\ g^N \end{pmatrix}, \qquad h = \begin{pmatrix} h^1 \\ \vdots \\ h^N \end{pmatrix}, \\ h^i(x,y) \in E^{p_i}, \qquad g^i(x,y) \in E^{m_i}, \qquad m_1 + \ldots + m_N = m, \\ h^i(x,y) = Y^i - \tilde{h}^i(x, Y^1, \ldots, Y^{i-1}, Y^{i+1}, \ldots, Y^N). \end{gathered} \tag{2.2}$$

The subvector X^i is the set of design variables corresponding to the ith subsystem. Similarly the subvector Y^i is the set of behavior variables of the ith subsystem. For any vector function $f(x,y)$, let $\hat{f}(X^i,Y^i)$ denote f with all the components $X^1, \ldots, X^{i-1}, X^{i+1}, \ldots, X^N, Y^1, \ldots, Y^{i-1}, Y^{i+1}, \ldots, Y^N$ fixed except for X^i and Y^i. Note the assumption that each Y^i can be explicitly determined in terms of x and the other subvectors Y^j.

2.2 Simple Sequential Decomposition.

The approach is to first divide the given large problem into a set of independent subproblems, corresponding naturally to the subsystems comprising the larger system. The ith subsystem would be

$$\begin{aligned} & \min_{X^i} \hat{\Theta}(X^i) \\ \text{subject to} \quad & \hat{g}^i(X^i,Y^i) \leq 0, \\ & \hat{h}^i(X^i,Y^i) = 0, \end{aligned} \tag{2.3}$$

where the system of equalities $\hat{h}^i = 0$ is used to eliminate Y^i from $\hat{g}^i$. The subproblems are solved sequentially for $i = 1,\ldots,N$, with one pass through all the subsystems constituting one outer iteration. The outer iterations are repeated until the same point $(\bar{x},\bar{y})$ solves all N subproblems. While solving the ith subsystem the values of $X^1,\ldots,X^{i-1},X^{i+1},\ldots,X^N$, $Y^1,\ldots,Y^{i-1},Y^{i+1},\ldots,Y^N$ are fixed. They can be chosen in a Gauss-Seidel manner where the first $i-1$ X and Y subvectors used have their latest values from solving the first $i-1$ subproblems. A parallel algorithm, solving the subproblems concurrently, would use a Jacobi scheme where the values of all the X^j and Y^j vectors are updated only at the end of each major outer iteration. The ensuing discussion of the current scheme assumes a Jacobi scheme.

3. DECOMPOSITION WITH APPROXIMATE COUPLING.

3.1 Cumulative Constraints.

In the scheme proposed by Sobieski [32], a measure of the constraints in each of the other subsystems is also brought into the ith subsystem in the form of one cumulative constraint C_i^k per subsystem. The approximate cumulative constraint C_i^k of the kth subsystem in the ith subsystem is obtained from the corresponding constraints $g^k \in E^{m_k}$ as a linearization of the Kreisselmeier-Steinhauser cumulative constraint

$$K_k(x,y) = \frac{1}{\rho} \ln \left(\sum_{j=1}^{m_k} e^{\rho g_j^k(x,y)} \right). \tag{3.1}$$

The ρ in the Kreisselmeier-Steinhauser function is a constant used to control the accuracy of the cumulative constraint approximation. The linearization of this cumulative constraint of the kth subsystem with respect to the variables of the ith subsystem is

$$C_i^k(X^i,Y^i) = \hat{K}_k(X_0^i,Y_0^i) + \sum_{j=1}^{n_i} \frac{\partial \hat{K}_k}{\partial X_j^i}(X_0^i,Y_0^i) \left(X_j^i - (X_0^i)_j \right). \tag{3.2}$$

3.2 Subspace Optimization Problem.

In the ith subsystem the cumulative constraints of the other subsystems are brought in as constraints. Therefore, a violated cumulative constraint of one subsystem may be satisfied by decisions taken in every one of the other subsystems. Therefore, we introduce coefficients r_i^p to represent the fractional "responsibility" assigned to the ith subsystem for reducing the violation of the cumulative constraint of the pth subsystem, for each $p = 1, \ldots, N$. Thus we have N^2 r-coefficients. The r_i^p's are defined in such a way that

$$\sum_{i=1}^{N} r_i^p = 1, \tag{3.3}$$

Sobieski [32] suggested the initialization of the r-coefficients in such a way that they are proportional to the degree of influence exerted by the ith subsystem on the pth cumulative constraint. This initialization is discussed in Appendix A.

To further reduce the objective function we allow cumulative constraints to be violated in one subsystem, provided that the violation will be offset by oversatisfaction of that constraint in another subsystem. To account for such tradeoffs, we introduce the N^2 coefficients t_i^p, corresponding to the cumulative constraint of the pth subsystem when present in the ith subsystem. For the pth cumulative constraint,

$$\sum_{i=1}^{N} t_i^p = 0, \tag{3.4}$$

maintains the constraint at a value of zero. This condition and the condition on the r-coefficients are enforced in what is called the coordination optimization phase, which is solved to update the values of the r's and the t's at the end of every outer iteration. The t_i^p's are initialized at the beginning of the algorithm to zero.

As has been described above, the r_i^p's are needed only in the case of a violation and the t_i^p's only when the constraints are critical, therefore only one of the two is needed at a time. Therefore we introduce N coefficients s^p which act as switches, one for each of the cumulative constraints of the subsystems. s^p is set to one (activating the r-coefficients) if the corresponding constraint $K_p \leq 0$ is violated at the outset of the system optimization procedure and stays at one until the K_p is driven to a critical status (zero value). Once K_p becomes critical, s^p is reset to zero (activating the t-coefficients) and stays at zero until the system optimization procedure terminates. The switch s^i is applied selectively to the natural constraints g^i of the ith subsystem (i.e., the constraints that are assigned to the ith subsystem) by multiplying the r-coefficient r_i^i by a factor of $\max\{\hat{g}^i(X_0^i, Y_0^i), 0\}$, so that constraints which are already satisfied are not taken into consideration.

Thus, the ith subsystem optimization problem is

$$\begin{aligned}
&\min_{X^i} \hat{\Theta}(X^i) \\
\text{subject to} \quad &\hat{g}^i(X^i, Y^i) \leq s^i \max\{\hat{g}^i(X_0^i, Y_0^i), 0\}(1 - r_i^i) + (1 - s^i)t_i^i, \\
&C_i^p(X^i, Y^i) \leq \hat{K}_p(X_0^i, Y_0^i)\, s^p(1 - r_i^p) + (1 - s^p)t_i^p, \qquad p = 1, \ldots, i-1, i+1, \ldots, N, \\
&\hat{h}^i(X^i, Y^i) = 0.
\end{aligned} \tag{3.5}$$

3.3 Coordination Optimization Phase.

The constrained minimum of Θ obtained from each subsystem optimization is a function of the constants r_i^p and t_i^p, and its partial derivatives with respect to r_i^p and t_i^p (assuming they exist) can be computed from the expressions given in Appendix A using gradient information for the Θ and C functions. These derivatives are used for a linear approximation of Θ that is the objective function for the *coordination optimization phase*, the last (and synchronizing) step of an outer iteration.

The coordination optimization phase (COP) solves a linear program to adjust the coefficients r_i^p and t_i^p, so that the objective function Θ will be further reduced (if possible) at the end of the next outer iteration. The linear program uses a linear extrapolation of Θ based on the partial derivatives $\partial\Theta/\partial z$ described above. Here z represents either an r or a t coefficient. Move limits (upper and lower bounds U_i^p, $\tilde{U}_i^p$, L_i^p and $\tilde{L}_i^p$ for r_i^p and t_i^p, respectively) are needed to prevent large changes in the r- and t-coefficients caused by the nonlinearity of the original problem. For the first COP execution, the r_i^p's may be initialized as already suggested and the t_i^p's are initialized to zero. For every subsequent execution, the r_i^p's and the t_i^p's are initialized to the terminal values from the previous COP execution. The result of the COP execution is a new set of r_i^p's and t_i^p's to be used in the next outer loop of subsystem optimizations. The adjustment of the r_i^p's and t_i^p's to the new values amounts to a reassignment of the responsibility for eliminating the constraint violations among the subsystems and to issuing a new set of instructions about trading the constraint violations/oversatisfactions among these subsystems. Let (x_0, y_0) be the current updated point (the result of the Jacobi outer iteration) and

$$\Theta_1 = \Theta(x_0, y_0) + \sum_{p=1}^{N}\sum_{i=1}^{N} \frac{\partial\Theta}{\partial r_i^p}\Delta r_i^p + \sum_{p=1}^{N}\sum_{i=1}^{N} \frac{\partial\Theta}{\partial t_i^p}\Delta t_i^p, \tag{3.6}$$

where $\Delta r_i^p = \left(r_i^p - (r_i^p)_0\right)$ and $\Delta t_i^p = \left(t_i^p - (t_i^p)_0\right)$. The partial derivatives $\partial\Theta/\partial r_i^p$ and $\partial\Theta/\partial t_i^p$ are evaluated at the optimal point computed by the ith subsystem optimization. Let

$$R = \left(r_1^1, r_1^2, \ldots, r_1^N, r_2^1, \ldots, r_2^N, \ldots, r_N^1, \ldots, r_N^N\right) \tag{3.7}$$

and

$$T = (t_1^1, t_1^2, \ldots, t_1^N, t_2^1, \ldots, t_2^N, \ldots, t_N^1, \ldots, t_N^N). \tag{3.8}$$

Then Θ_1 is a function of R and T. The linear program solved during the coordination optimization phase is:

$$\begin{aligned}
&\min_{R,T} \Theta_1(R,T) \\
\text{subject to} \quad &\sum_{k=1}^{N} r_k^p = 1, \qquad p = 1, \ldots, N, \\
&\sum_{k=1}^{N} t_k^p = 0, \qquad p = 1, \ldots, N, \\
&0 \le r_k^p \le 1, \qquad p = 1, \ldots, N, \quad k = 1, \ldots, N, \\
&L_k^p \le r_k^p \le U_k^p, \qquad p = 1, \ldots, N, \quad k = 1, \ldots, N, \\
&\tilde{L}_k^p \le t_k^p \le \tilde{U}_k^p, \qquad p = 1, \ldots, N, \quad k = 1, \ldots, N.
\end{aligned} \tag{3.9}$$

4. PSEUDOCODE FOR ALGORITHM.

An algorithmic description of the whole process in pseudo-code is given next, using the following model quadratic programming problem (without the variables y and equality constraints $h(x,y)=0$) for specificity:

$$\begin{aligned} &\min_x \ x^t A x \\ &\text{subject to} \quad Bx \leq d, \end{aligned} \tag{4.1}$$

where

$$A = \begin{pmatrix} A_{11} & \alpha_{12}A_{12} & \dots & \alpha_{1N}A_{1N} \\ \alpha_{12}A_{12}^t & A_{22} & \dots & \alpha_{2N}A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1N}A_{1N}^t & \alpha_{2N}A_{2N}^t & \dots & A_{NN} \end{pmatrix},$$

$$B = \begin{pmatrix} B_{11} & \beta_{12}B_{12} & \dots & \beta_{1N}B_{1N} \\ \beta_{21}B_{21} & B_{22} & \dots & \beta_{2N}B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1}B_{N1} & \beta_{N2}B_{N2} & \dots & B_{NN} \end{pmatrix},$$

$$d = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}, \qquad x = \begin{pmatrix} X^1 \\ X^2 \\ \vdots \\ X^N \end{pmatrix},$$

and $X^i \in E^{n_i}$, $A_{ij} \in E^{n_i \times n_j}$, $B_{ij} \in E^{p_i \times n_j}$, $d_i \in E^{p_i}$, the A_{ii} are symmetric and positive definite and the α_{ij}, β_{ij} are fixed "coupling" parameters, for all $i,j = 1,\dots,N$. Using notation defined in Appendix A, pseudo-code for the algorithm applied to this quadratic programming problem (QP) is:

Choose an initial estimate x and initialize the r-, s- and t-coefficients;

Repeat until minimum reached

begin

  for $i = 1$ **to** N **do**

  begin

    Calculate the linearization $C_i^j(X^i)$ of the cumulative constraint for the jth subsystem, for all $j \neq i$;

Calculate the ith subsystem's self responsibility

$$\delta^i = s^i \max\left\{\hat{g}^i\left(X_0^i, Y_0^i\right), 0\right\}\left(1 - r_i^i\right) + \left(1 - s^i\right) t_i^i;$$

Solve the QP (ith subsystem)

$$\min_{X^i} \hat{\Theta}\left(X^i\right) = (X^i)^t A_{ii} X^i + 2\left(\sum_{j \neq i} \alpha_{ij} (X^i)^t A_{ij} X^j\right)$$

$$\text{subject to} \quad \sum_{j=1}^{N} \beta_{ij} B_{ij} X^j - d_i \leq \delta^i, \qquad (\beta_{ii} = 1)$$

$$\tilde{C}_i^j\left(X^i\right) \leq 0, \qquad \text{for all } j \neq i;$$

Calculate (if not already available) the Lagrange multipliers λ using the method given in Appendix A.

Calculate $\dfrac{\partial \Theta}{\partial r_i^j}$ and $\dfrac{\partial \Theta}{\partial t_i^j}$ for $j = 1, \ldots, N$;

end

Solve the LP (Coordination Optimization Phase)

$$\min_{R,T} \Theta_1(R, T)$$

$$\text{subject to} \quad \sum_{k=1}^{N} r_k^p = 1, \qquad \sum_{k=1}^{N} t_k^p = 0, \qquad p = 1, \ldots, N,$$

$$0 \leq r_k^p \leq 1, \qquad L_k^p \leq r_k^p \leq U_k^p, \qquad \tilde{L}_k^p \leq t_k^p \leq \tilde{U}_k^p,$$

$$p = 1, \ldots, N, \qquad k = 1, \ldots, N;$$

end (repeat)

5. INITIAL TESTS.

5.1 2×2 Example.

Testing of this algorithm was first performed on a simple 2×2 case:

Example 1.

$$\begin{aligned} \min_{x} \quad & x_1^2 + x_2^2 \\ \text{subject to} \quad & x_1 + \beta x_2 \leq 4, \\ & \beta x_1 + x_2 \geq 2, \\ & \text{where } x = (x_1, x_2)^t \in E^2. \end{aligned} \tag{5.1}$$

Here each constraint is taken to be in a subsystem by itself with $X^1 = (x_1)$ and $X^2 = (x_2)$. The results are tabulated in Table I. The column headings are the starting points, the last column gives the solutions for the different values of β, and each entry contains a convergence code and the number of iterations taken. The code IF means an infeasible subproblem is encountered at the very first iteration and the procedure is terminated, R means the solution is reached in the iteration indicated, but subsequently an infeasible subproblem is encountered, WR means a wrong point is reached before an infeasible subproblem causes termination, O means there is oscillation through the number of iterations indicated, WC means there is convergence to a point other than the solution, and NC means there is no convergence even after the number of iterations indicated.

TABLE I
Original algorithm applied to the 2×2 *case.*

β	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	solution
0.0	R 1	R 1	R 1	R 1	R 1	(0.0,2.0)
0.1	WR 1	IF 1	WC 4	WC 4	IF 1	(0.198,1.98)
0.3	WR 1	IF 1	WC 5	WC 4	IF 1	(0.55,1.835)
0.5	WR 1	IF 1	O 150	NC 150	WR 1	(0.8,1.6)
1.0	O 150	O 150	O 150	O 150	O 150	(1.0,1.0)

As can be seen from the table, the main problem with the algorithm is not being able to deal with infeasibility in a subproblem. This issue of infeasibility was mentioned in the appendix of [33], and is addressed in the next section. Another reason for the algorithm not being successful is the way the s-coefficient is set permanently to zero once a constraint becomes critical. When a constraint that was once critical becomes violated, the r-coefficients cannot be brought in to reduce the violation.

6. MODIFICATIONS TO THE ORIGINAL ALGORITHM.

Several modifications and variations of the original algorithm as described in [32] are discussed next. The order of the topics is not significant.

Natural Constraints.

The original algorithm, while solving a particular subsystem, uses a cumulative constraint to represent the constraints of each of the subsystems, including those of the subsystem being solved. In the modified version we use a cumulative constraint for each of the other subsystems, but the natural constraints of the subsystem being solved are used instead of the cumulative constraint representing them. The algorithm described in Section 4 includes this modification.

Changes in setting of the switch coefficients s^p.

When a K_p becomes critical, the corresponding s^p is set to zero and stays at zero until the whole procedure terminates. This means that the term with the r^p coefficient does not contribute any longer to the constraints. If the constraint becomes violated later, then the violation cannot be reduced using the r-coefficients. Hence two alternatives to the algorithm were considered. One was to remove the s-coefficient from the r term. The other was to set the s^p coefficient at the end of every outer iteration depending on whether the corresponding constraint was satisfied or not. This would make one of the r or t-coefficients active all the time. The second alternative performed better in initial tests and therefore was selected.

Handling infeasibility.

Because of the linearization and the allocation of responsibility, some subproblems may be infeasible. The following procedure is employed to recover from infeasibility in a subsystem. A new variable ω is introduced in each of the constraints and a large multiple of this variable is added to the objective function to be minimized. Thus, the corresponding subproblem would now be

$$\begin{aligned} &\min_{X^i} \hat{\Theta}(X^i) + M\omega \\ \text{subject to} \quad &\hat{g}^i(X^i, Y^i) - \omega \le s^i \max\{\hat{g}^i(X_0^i, Y_0^i), 0\}(1 - r_i^i) + (1 - s^i)t_i^i, \\ &C_i^p(X^i, Y^i) - \omega \le \hat{K}_p(X_0^i, Y_0^i)\, s^p(1 - r_i^p) + (1 - s^p)t_i^p, \\ &\qquad\qquad p = 1, \ldots, i-1, i+1, \ldots, N, \end{aligned} \tag{6.1}$$

where M is a large positive number.

The linear objective function in the Coordination Optimization Phase is formed using the Lagrange multipliers obtained from the subproblems. The introduction of the variable ω affects these Lagrange multipliers. The sensitivity derivatives are affected considerably because of these Lagrange multipliers, as indicated in Appendix A. A thought as to whether this was justified or not led to two variations of the algorithm. In the first variation, in case of an infeasibility in any subproblem, the Coordination Optimization Phase is omitted at the end of that outer iteration. In the second variation, the COP is included in every outer iteration.

Limit on t-coefficients.

Initially tests were performed with the t coefficients left unbounded, but clearly this is unwise. A few variations for the bounds on the t's were considered. One possibility is keeping the bound fixed throughout the procedure, but this may result in nonconvergence to the solution. Also the bounds should not decrease too fast, because this may force convergence to a nonoptimal solution.

The bound is reduced by a factor of $f = 0.8$ at the end of every outer iteration. Thus, if the bound at the first iteration is t_1, then the bound at the mth iteration is

$$t_m = 0.8^{(m-1)} t_1. \tag{6.2}$$

A variation of having the bound at the mth iteration equal a factor $f^{log(m-1)}$ or a factor $f^{(m-1)^{1/2}}$ of the bound at the first iteration was also considered. Bloebaum, Hajela, and Sobieski [7] also experimented with variable limits on the t-coefficients.

A later modification was to change the move limits on the t-coefficients based on information about the corresponding coefficients in the objective function of the COP. To ensure that no subsystem is allowed a violation that cannot be offset by an equivalent oversatisfaction in the other subsystems, a change was made to the move limits on the t-coefficients, using information about the corresponding sensitivities. The a^{pk} coefficient (as described in Appendix A) is a measure of the sensitivity of the pth cumulative constraint to

the variables of the kth subsystem. The lower limit of t_k^p is now $\max\{-a^{pk}, -t_m\}$, reasoning that the oversatisfaction expected of the pth cumulative constraint in the kth subsystem will be restricted to what it can handle. The a^{pk} coefficients are obtained at every iteration using the current value of the x vector.

The initial value of the move limit on the t-coefficient affects the results and the path taken. Various values of this initial limit were tried. Another alternative to starting with a larger limit on t is to fix the limit on the t-coefficient to some smaller value for a few iterations, and then perform the reduction as described earlier. This is to make sure that after a certain number of iterations, the t-coefficient can take on a value large enough to contribute to the solution process.

At the beginning of every major iteration involving a new ρ, the initial value of the move limit on the t-coefficient may be reset to either the original value or some fraction of it.

Convergence criterion.

The convergence criterion initially involved a measure of the difference between three successive iteration values of the design vector. Later, tests revealed that with this criterion the procedure could stop even if there was a chance for further improvement, because of changes in the values of the r and t-coefficients. Hence the difference between the values of the t and r-coefficients were also included in the convergence criterion. Thus, if S_{m-2} represents the normalized form of the vector (x, R, T) at the $(m-2)$th iteration and similarly for S_{m-1} and S_m, then using the 2-norm the convergence criterion is

$$\|S_{m-1} - S_{m-2}\| + \|S_m - S_{m-1}\| \leq 0.0001. \tag{6.3}$$

Changes to the ρ coefficient.

After the required convergence criterion is met the ρ coefficient is increased and the whole process is repeated again to check if the convergence criterion is still met; if not another major iteration is performed. This is because with increasing ρ the cumulative constraint is closer to the actual constraints. The process is not started with a large ρ, as the problem is then very ill conditioned.

Cross derivatives.

The cross derivatives are checked to see if one subsystem is at all dependent on the variables of another subsystem, if not the corresponding r and t-coefficients are fixed at zero. This is done at the end of every outer iteration.

Changes to the r-coefficients.

The diagonal r-coefficients r_i^i are assigned a minimum value of 0.2 always, reasoning that every subsystem always has some responsibility towards its own constraints. In addition, a 20% move limit is imposed on all the r-coefficients. Thus if r_k^p is the value of an r-coefficient in the nth iteration and $\tilde{r}_k^p$ is the value of the same r-coefficient in the $(n-1)$st iteration, then

$$0.8\tilde{r}_k^p \leq r_k^p \leq 1.2\tilde{r}_k^p + 0.01 \tag{6.4}$$

No COP.

If the objective function of the COP is a constant then the COP is skipped for that iteration, which prevents the possibility of the t-coefficients being assigned arbitrary values.

Resetting the t-coefficients.

It was observed in one case that t-coefficients (corresponding to one subsystem's constraint) with equal derivatives in the objective function of the COP assigned extreme values to the corresponding t-coefficients even though their combined contribution to the objective function was zero. Setting each of these t-coefficients to zero will also keep their contribution to the objective function at zero. Hence, after the COP a check is performed on the t-coefficients to see if for a particular p the sum of the contributions of all the corresponding coefficients to the objective function of the COP is zero. If so all the t coefficients corresponding to this p are forced to be zero. This check is performed for all values of p.

Different combinations of these modifications were used on three different test problems and the results are given in Tables II-VII in Chapter 7.

7. FURTHER TESTS.

7.1 Modified algorithm on 2×2 Example.

The original algorithm as proposed by Sobieszczanski-Sobieski [32] did not prove to be successful as indicated by Table I. Results of two of the most successful variations to this algorithm tested on the 2×2 case are tabulated in Tables II and III. The characteristics of the algorithms used are given above the corresponding tables. "s updated" indicates that the s- coefficient is updated at the end of every outer iteration as indicated in the modifications given in Section 7. "ω used" means that an artificial variable ω was introduced to deal with infeasible subproblems. The inclusion or exclusion of the COP is in the case of an infeasibility in any subproblem. The limit on the magnitude of the t-coefficient is 1 initially and this bound is decreased using a factor of 0.8 as described in Section 7. The most successful version was used for larger test problems like the 3×3 case with two subsystems and the 6×6 case with three subsystems. The tests were carried out for five different values of β and for five different starting points. The column headings are the starting points, the last column gives the solutions for the different values of β, and each entry contains a convergence code, and the number of iterations until the two-norm of the change in (x, R, T) is less than 0.0001. For Tables II and III the number of iterations the limit on the t-coefficient is held fixed (at 1.0) before being reduced is 10, and 30 for Table IV. The code C means there is convergence to the solution, WC means there is convergence but not to the solution, and NC means there is no convergence even in the specified number of iterations.

TABLE II
s updated, ω used, no COP,
t bound at 1 and 0.8 update
after 10 iterations.

β	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	solution
0.0	C 6	C 6	C 6	C 6	C 6	(0.0,2.0)
0.1	C 71	WC 12	C 71	C 71	C 71	(0.198,1.98)
0.3	C 67	WC 25	C 64	C 67	C 67	(0.55,1.835)
0.5	C 66	WC 18	C 66	C 65	C 64	(0.8,1.6)
1.0	C 7	C 8	C 7	C 62	C 8	(1.0,1.0)

TABLE III
s updated, ω used, COP,
t bound at 1 and 0.8 update
after 10 iterations.

β	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	solution
0.0	C 6	C 6	C 6	C 6	C 6	(0.0,2.0)
0.1	C 71	C 67	C 71	C 71	C 71	(0.198,1.98)
0.3	C 67	C 72	C 64	C 67	C 67	(0.55,1.835)
0.5	C 66	C 65	C 66	C 65	C 64	(0.8,1.6)
1.0	C 7	C 8	C 7	C 62	C 8	(1.0,1.0)

Figures 1 and 2 are provided for a better understanding of the path taken from the starting point to the solution. The pictures correspond to Example 1 with $\beta = 0.1$. Figure 1 includes all the iterate values (except for the starting point) up to the solution. The first few segments are numbered $1 - 8$ at their midpoints. Figure 2 is a blown up view of the region of convergence that is marked in Figure 1. Some of the intermediate segments are numbered at their midpoints here. The solution is indicated by a $*$.

7.2 3×3 Example.

Example 2.

$$\begin{aligned} \min_x \quad & x_1^2 + x_2^2 + x_3^2 \\ \text{subject to} \quad & x_1 + x_2 + \beta x_3 \leq 4, \\ & -x_1 - x_2 - \beta x_3 \leq -2, \\ & -\beta x_1 - \beta x_2 - 5x_3 \leq -2, \\ & \text{where } x = (x_1, x_2, x_3)^t \in E^3 \end{aligned} \tag{7.1}$$

Here, the first two constraints belong to one subsystem and the third constraint to another subsystem, $X^1 = (x_1, x_2)$ and $X^2 = (x_3)$.

In Tables II–IV for $\beta = 0$, the 6 iterations corresponds to 3 iterations for the convergence test to be satisfied with 2 consecutive values of ρ; thus 6 actually is the smallest possible

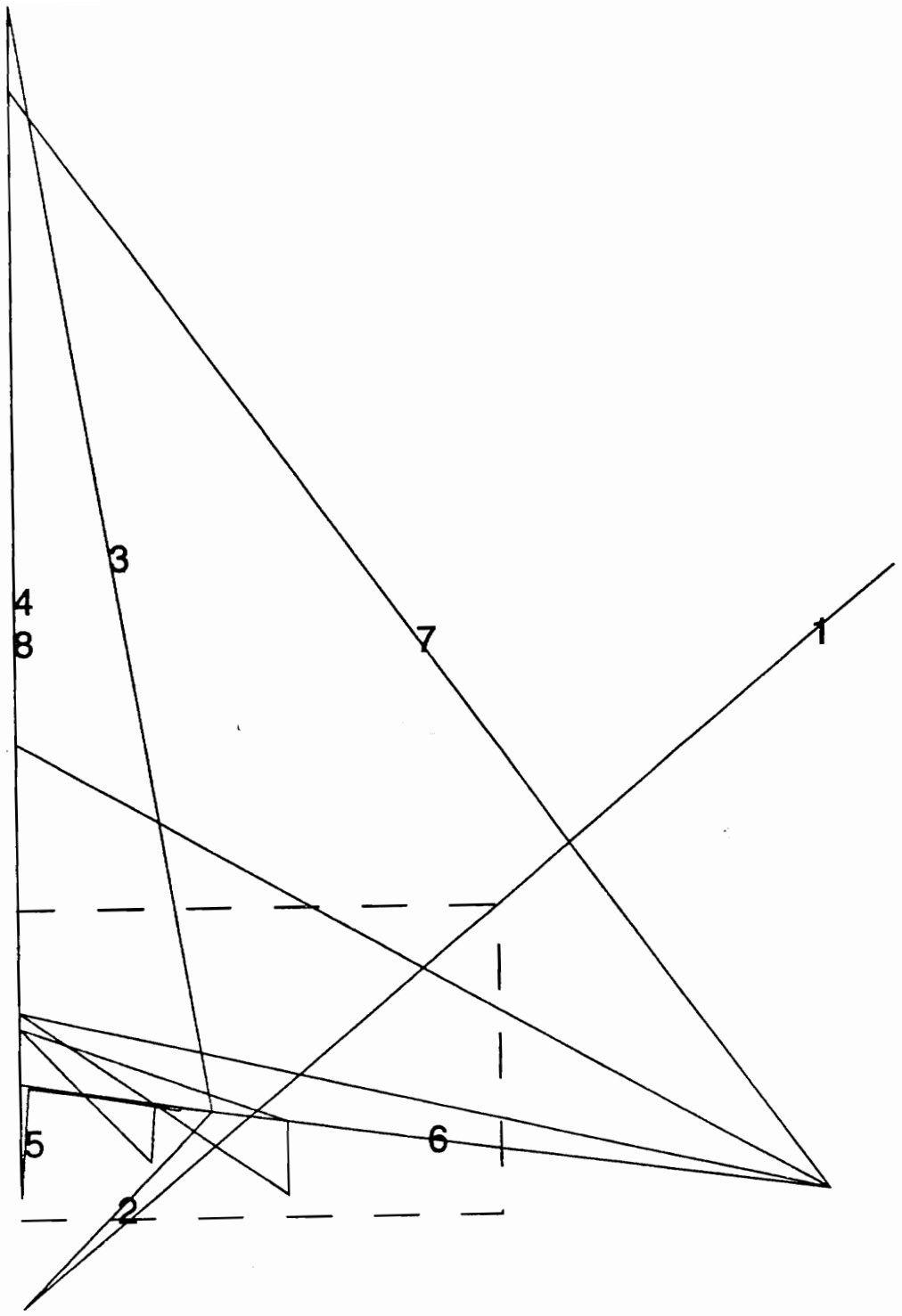

FIGURE 1. *Trace of the solution iterates for Example 1, corresponding to* $\beta = 0.1$, *with starting point (2,3) and solution (0.198,1.98).*

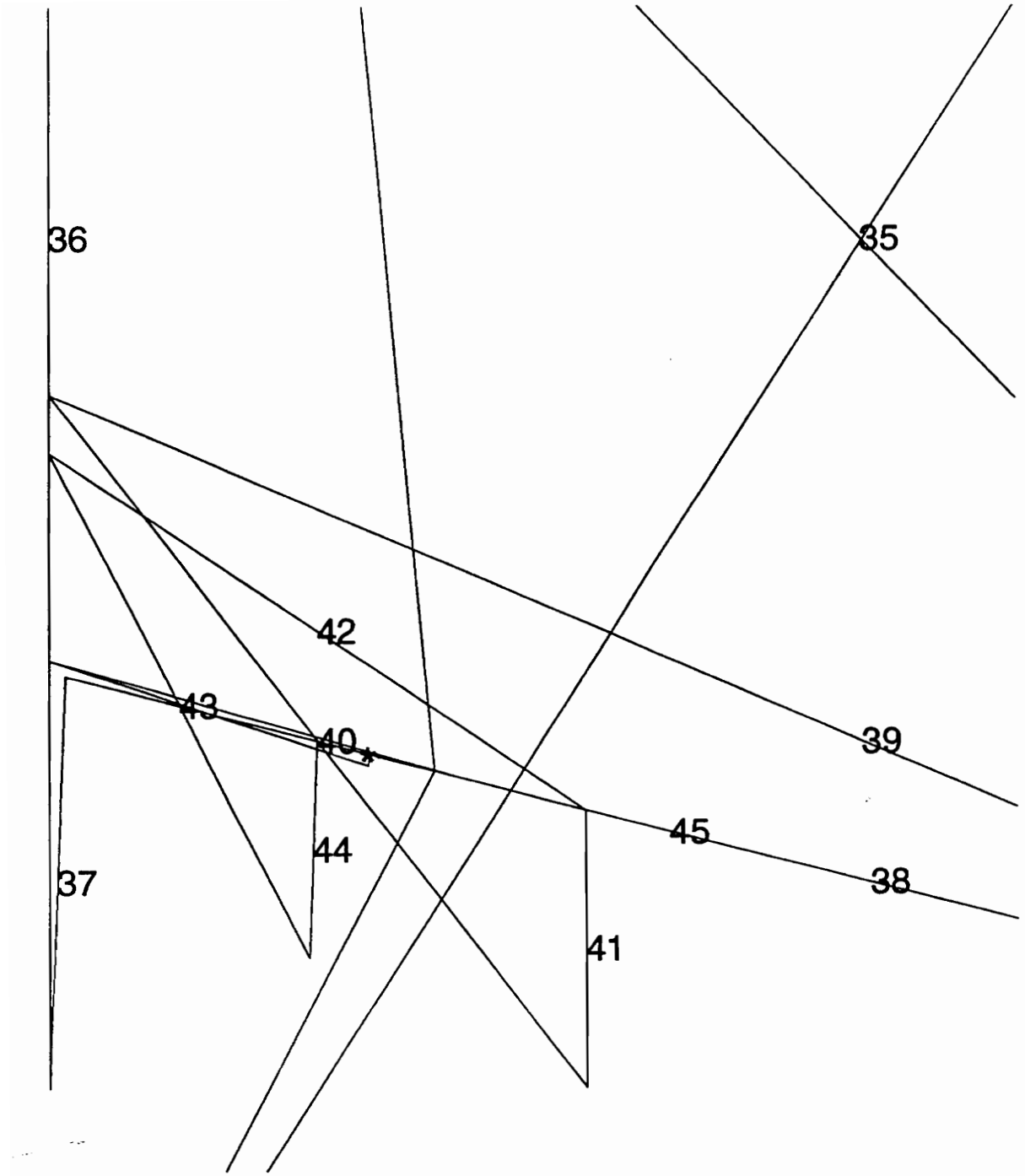

FIGURE 2. *Blown up view of the region of convergence of the plot shown in Figure 1.*

TABLE IV
s updated, ω used, COP,
t bound at 1 and 0.8 update
after 30 iterations.

β	(0,1,-3)		(1,1,0)		(4,0.1,0.8)		(-10,3,-10)		(0,0,0)		solution
0.0	C	6	C	6	C	6	C	6	C	6	(1,1,0.4)
0.1	C	86	C	89	C	98	C	89	C	87	(0.9819,0.9819,0.3607)
0.3	C	85	C	82	C	92	C	82	C	80	(0.9569,0.9569,0.2870)
0.5	C	93	C	81	C	81	C	89	WC	80	(0.8888,0.8888,0.4444)
1.0	WC	102	C	103	C	77	WC	102	C	76	(0.6666,0.6666,0.6666)

number of iterations, and corresponds to reaching the solution in one step. For the WC entries in Table IV, performing a cold start at the point they converge to results in convergence to the correct solution. Figure 3 corresponds to Example 2 and gives a trace of the iterates for $\beta = 0.5$ and starting point $(0, 1, -3)$ (not included in the picture).

Move limits on the x vector.

Introducing move limits on the x variables will help prevent oscillation of the iterates. The size of the move limits permitted can be varied depending on the problem. Let m be the move limit permitted and $\bar{x}_i$ the current value of x_i. Then the constraint

$$\bar{x}_i - m|\bar{x}_i| - 0.1 \leq x_i \leq \bar{x}_i + m|\bar{x}_i| + 0.1 \tag{7.2}$$

is added for every component x_i of the x vector.

Tests were performed on the following 6×6 example using different move limits and the results are tabulated accordingly.

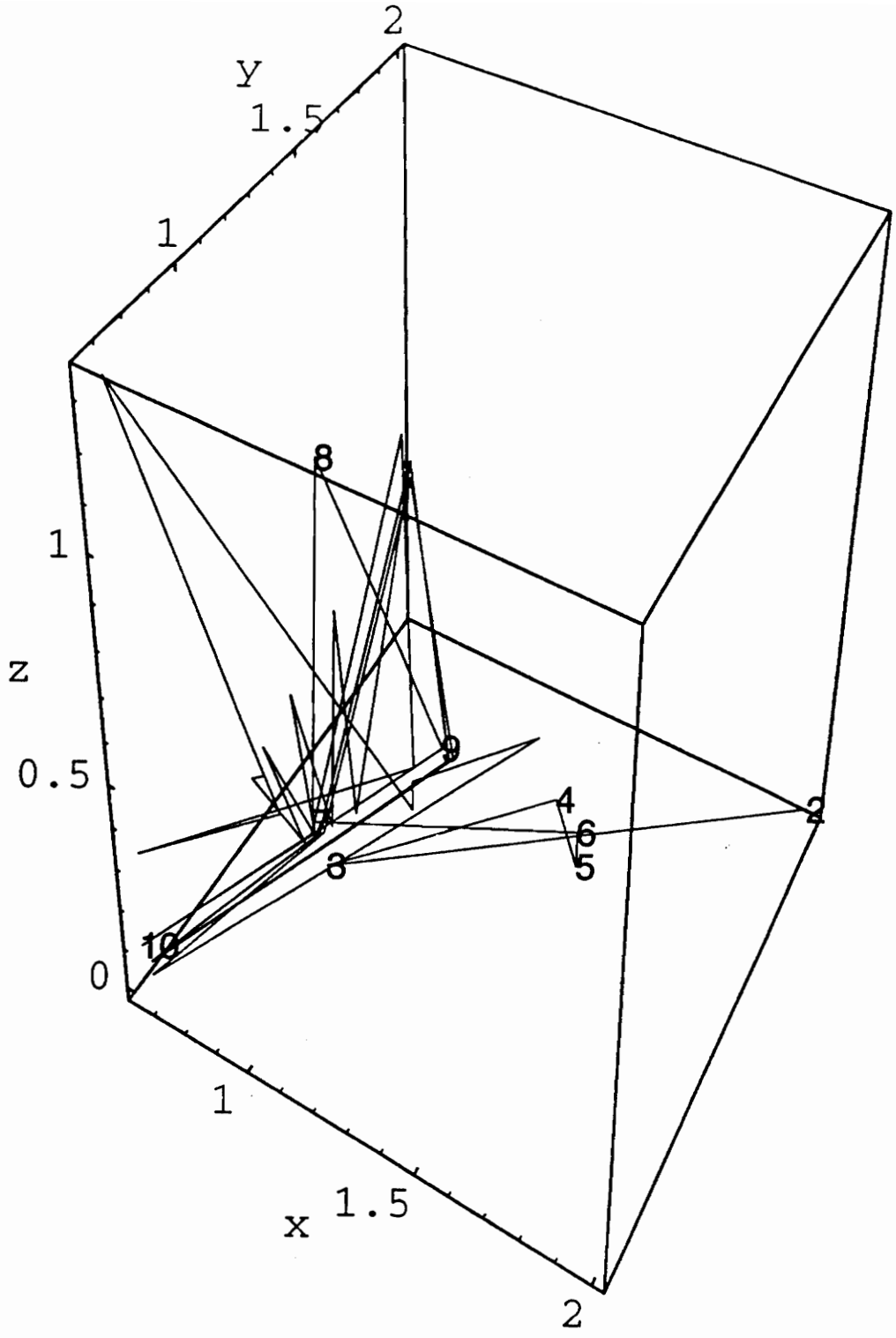

FIGURE 3. *Trace of the solution iterates for Example 2, corresponding to* $\beta = 0.5$, *with starting point (0,1,-3) and solution (0.88,0.88,0.44).*

7.3 6×6 **Example.**

Example 3.

$$
\begin{aligned}
\min_{x} \quad & x_1^2 + x_2^2 + x_3^2 + 2.5x_4^2 + 2.5x_5^2 + 10x_6^2 \\
\text{subject to} \quad & x_1 + x_2 + x_3 + 0 - \beta x_5 - 2\beta x_6 \leq 4, \\
& -x_1 - x_2 - x_3 - \beta x_4 + 0 + 0 \leq -2, \\
& -x_1 - x_2 - 5x_3 + 0 + 0 + 0 \leq -2, \\
& 0 + 0 + 0 + x_4 + x_5 - \beta x_6 \leq -4, \\
& \beta x_1 + \beta x_2 + 0 - 5x_4 - 4x_5 - \beta x_6 \leq 20, \\
& \beta x_1 + \beta x_2 - \beta x_3 + 0 + 0 - x_6 \leq -6, \\
& \text{where } x = (x_1, x_2, x_3, x_4, x_5, x_6)^t \in E^6
\end{aligned}
\tag{7.3}
$$

Here there are three subsystems with $n_1 = 3$, $n_2 = 2$ and $n_3 = 1$.

If the algorithm as described thus far is used, the results are not encouraging as indicated by Table V. In Table V the number in each entry gives the number of iterations taken. The entries in this table correspond to a move limit of $m = 0.3$ on each component of the x vector. The limit on the t-coefficients is held fixed for 30 iterations before being reduced. There is convergence to the solution only in the case of $\beta = 0.0$; for larger βs the solution is not obtained for any of the starting points. An attempt to use the actual solution vector as the starting point also did not lead to convergence to the solution – the process actually diverged away from the solution. Clearly, this is a fundamental flaw.

The cause of the divergence was traced to the sensitivity information used in the COP. This would therefore affect the r- and t-coefficients that are updated in the COP. The coefficients in the linear objective function of the COP are the partial derivatives of the main objective function (of the global problem) Θ with respect to each of the r- and the t-coefficients. Consider one such derivative $\partial\Theta/\partial z$, where z represents any one of the r- or the t-coefficients. It is calculated using the Lagrange multipliers obtained from the subsystem optimizations (as given in Appendix A) via

$$
\frac{\partial \Theta}{\partial z} = \lambda^T \frac{\partial G_A^i}{\partial z}. \tag{7.4}
$$

The Lagrange multipliers obtained from the subsystems correspond to different values of the design vector x by necessity, because the subsystem optimizations must be done *independently* and *concurrently*. On completion of the subsystem optimizations, the x vector is updated as a composite of the optimal subvectors obtained from the subsystems. Thus, the sensitivity derivatives obtained using these Lagrange multipliers no longer reflect the actual sensitivity of the objective function at the updated x to changes in the r- and the t-coefficients.

To illustrate this point, consider the following step in the optimization process for Example 3 with $\beta = 0.1$ and the solution vector $(-2.4, -2.4, 7.0, -1.7, -1.8, 4.8)$ as the starting point x_0. At the end of the first outer iteration the value of the design vector is

$$x_1 = (-2.4, -2.4, 6.99, -1.7, -1.7, 4.8).$$

The derivatives of the objective function Θ with respect to the t-coefficients corresponding to the 2nd constraint (i.e., t_1^2, t_2^2 and t_3^2) are calculated to be 0.0, -8.7, and 0.0 respectively, and at the end of the COP these t-coefficients are assigned the values 0.0, 0.1, and -0.1 respectively.

If the sensitivity information were correct, the derivative of Θ with respect to t_3^2 being zero should ensure that the objective function is not sensitive to the value of t_3^2. In the third subsystem optimization, the cumulative constraint $\tilde{C}_3^2 \leq 0$ corresponding to the second subsystem is active at the solution, thus determining the value of the variable x_6. This constraint is $0.1x_6 \geq 0.58$. Therefore $x_6 = 5.8$, which considerably increases the value of the objective function. If the value of t_3^2 had been 0.0 instead of the -0.1 it was assigned, the constraint $\tilde{C}_3^2 \leq 0$ would have been $0.1x_6 \geq 0.48$ and x_6 would have been unaltered from its value 4.8 at the end of the previous iteration. Thus, the objective function is clearly very sensitive to the value of t_3^2. The objective function of the COP and the third subproblem in the second iteration are given in full below for a better understanding of the problem:

The objective function of the COP at the end of the first iteration is

$$\min_x \quad 0.027r_1^1 - 94t_1^3 - 8.7t_2^2 - 96.3t_3^3$$

The third subsystem is

$$\begin{aligned} \min_{x_6} \quad & x_6 \\ \text{subject to} \quad & x_6 \geq 4.7, \\ & 0.0x_6 \geq 0.0, \\ & 0.1x_6 \geq 0.58, \\ & 6.36 \geq x_6 \geq 3.27, \end{aligned} \tag{7.5}$$

Here, there is a move limit of ($m = 0.3$) on x_6, which gives the last constraint. The first constraint is $\tilde{g}_1^3 \leq 0$, the second constraint corresponds to $\tilde{C}_3^1 \leq 0$, and the third constraint corresponds to $\tilde{C}_3^2 \leq 0$.

Table VI gives the results for the 6×6 problem if the sensitivities used in the COP are the average of the sensitivities obtained over two iterations. In each entry in Tables VI-VII, the number to the right of the convergence code indicates the number of iterations taken, the number below gives the number of iterations the limit on the t-coefficient is held fixed (at 1.0) before being reduced, and the third number gives the move limit permitted on each component of the x vector. While smoothed sensitivities clearly help (compare Tables V and VI), they are not a cure for the larger β cases. We remark that simply changing the move limits on x, r, and t is not the remedy, since hundreds of move limit variations were tried with results no better than those in Table VI.

As a final illustration of the erratic behavior induced by the COP, we describe an irrational scheme resulting from a programming error. In this version, the a_k^p coefficients and the cross derivatives are not updated at the end of every outer iteration as described in the list of modifications given earlier in Section 7. Also, the a_k^p coefficients that are obtained initially are assigned incorrectly (i.e., a_k^p is taken to be a_p^k). The absolute value of 0.1 is not added to the move limit on the r-coefficients. For the move limit on the components of x the absolute value added is equal to the move limit. This irrational scheme applied to Example 3 yields Table VII, which is comparable to the results obtained by any of the more plausible versions on Example 3.

TABLE V
s updated, ω used, COP,
t bounded at 1 and 0.8 update.

β	(0,0,0, 0,0,0)	(1,2,3, -1,1,5)	(-10,4,4, 0.8,0.1,1)	(1,1,1, 1,1,1)	(-4,2,2, 0,1,1)	solution
0.0	C 17	C 17	C 19	C 17	C 17	(0.6,0.6,0.6,-2.0,-2.0,6.0)
0.1	NC 154	NC 154	WC 113	NC 154	NC 151	(-2.4,-2.4,7.0,-1.7,-1.8,4.8)
0.3	NC 151	WC 153	WC 105	WC 105	NC 150	(-2.7,-2.7,8.0,-1.5,-1.8,1.9)
0.5	WC 104	WC 197	WC 104	WC 106	WC 106	(-1.7,-1.7,6.3,-1.5,-1.9,1.0)
1.0	WC 102	WC 105	WC 102	WC 102	WC 105	(-0.5,-0.5,4.2,-1.2,-2.0,0.7)

TABLE VI
s updated, ω used, COP,
t bounded at 1 and 0.8 update.

β	(0,0,0, 0,0,0)	(1,2,3, -1,1,5)	(-10,4,4, 0.8,0.1,1)	(1,1,1, 1,1,1)	(-4,2,2, 0,1,1)	solution
0.0	C 17 10 0.3	C 17 10 0.3	C 19 10 0.3	C 17 10 0.3	C 17 10 0.3	(0.6,0.6,0.6,-2.0,-2.0,6.0)
0.1	C 110 30 0.1	C 190 70 0.3	C 113 30 0.1	C 72 10 0.3	WC 109 30 0.1	(-2.4,-2.4,7.0,-1.7,-1.8,4.8)
0.3	WC 105 30 0.1	C 203 80 0.1	C 103 30 0.1	C 163 60 0.1	WC 103 30 0.1	(-2.7,-2.7,8.0,-1.5,-1.8,1.9)
0.5	WC 103 30 0.1	WC 103 30 0.1	WC 104 30 0.1	WC 107 30 0.1	WC 107 30 0.1	(-1.7,-1.7,6.3,-1.5,-1.9,1.0)
1.0	WC 103 30 0.1	WC 103 30 0.1	WC 104 30 0.1	WC 102 30 0.1	WC 102 30 0.1	(-0.5,-0.5,4.2,-1.2,-2.0,0.7)

TABLE VII
s updated, ω used, COP,
t bound at 1 and 0.8 update.

β	(0,0,0, 0,0,0)		(1,2,3, -1,1,5)		(-10,4,4, 0.8,0.1,1)		(1,1,1, 1,1,1)		(-4,2,2, 0,1,1)		solution
0.0	C	16	C	14	C	19	C	14	C	15	(0.6,0.6,0.6,-2.0,-2.0,6.0)
	1	0.2	1	0.2	1	0.2	1	0.2	1	0.2	
0.1	WC	82	C	102	WC	23	C	102	WC	26	(-2.4,-2.4,7.0,-1.7,-1.8,4.8)
	30	0.3	30	0.1	30	0.3	30	0.1	30	0.3	
0.3	WC	34	C	199	WC	81	WC	101	WC	35	(-2.7,-2.7,8.0,-1.5,-1.8,1.9)
	30	0.3	80	0.5	30	0.3	30	0.3	30	0.3	
0.5	WC	26	C	200	WC	24	WC	110	WC	12	(-1.7,-1.7,6.3,-1.5,-1.9,1.0)
	30	0.3	80	0.8	30	0.3	30	0.3	30	0.3	
1.0	WC	20	WC	8	WC	20	WC	99	WC	15	(-0.5,-0.5,4.2,-1.2,-2.0,0.7)
	30	0.3	30	0.3	30	0.3	30	0.3	30	0.3	

8. QUADRATIC PROGRAMMING ALGORITHMS.

The proposed algorithm was tested on simple quadratic programming (QP) problems. When a QP problem is divided into subsystems, each subproblem is also a QP problem. This chapter gives the algorithm used to solve a QP subproblem. This algorithm by Fletcher [12] uses an active set strategy.

As mentioned before, an optimization package called QPSOL [16] was used extensively to verify results and the LPSOL part of this package was used to solve the linear programming problem that arises in the COP.

Also, another optimization package, MINOS, was used during the initial tests to verify the results obtained. The algorithms used in QPSOL and MINOS are given in Sections 8.2 and 8.3.

8.1 Active set strategy - Fletcher.

Let $x \in \mathbf{R}^n$ and $G \in \mathbf{R}^{n \times n}$ be positive definite. We need to find a solution x^* to a quadratic programming problem of the following type:

$$\begin{aligned} &\underset{x}{\text{minimize}}\, q(x) \equiv \frac{1}{2} x^T G x + g^T x, \\ &\text{subject to } a_i^T x = b_i, \qquad i \in E, \\ &\qquad\qquad a_i^T x \geq b_i, \qquad i \in I. \end{aligned} \tag{8.1}$$

The implemented algorithm is as follows: First, the QP problem with just equality constraints is solved by one of two different methods. Next these methods are generalized to handle inequality constraints also by means of an active set method.

In the active set method [12], certain constraints, indexed by the active set $\mathcal{A}$, are regarded as equalities while the rest are temporarily disregarded, and the method adjusts this set in order to identify the correct active constraints at the solution to the given QP problem.

On iteration k, a feasible point $x^{(k)}$ is known which satisfies the active constraints as equalities, that is, $a_i^T x^{(k)} = b_i$, $i \in \mathcal{A}$. Each iteration attempts to locate the solution to an

equality problem (EP) in which only the active constraints occur. This is done by shifting the origin to $x^{(k)}$ and looking for a correction $\delta^{(k)}$ which solves

$$\begin{aligned} &\underset{\delta}{\text{minimize}} \ \frac{1}{2}\delta^T G\delta + \delta^T g_{(k)}, \\ &\text{subject to } a_i^T\delta = 0, \qquad i \in \mathcal{A}, \end{aligned} \tag{8.2}$$

where $g^{(k)}$ is defined by $g^{(k)} = g + Gx^{(k)}$ and is $\nabla q(x^{(k)})$ for the function $q(x)$ defined before. This is in the form of an EP and can be solved by one of the methods given below.

If the step $\delta^{(k)}$ is feasible with regard to the constraints not in $\mathcal{A}$, then the next iterate is taken as $x^{(k+1)} = x^{(k)} + \delta^{(k)}$. If not, a line search is made in the direction of $\delta^{(k)}$ to find the best feasible point. This is done by first defining the solution of the EP for the correction to be $s^{(k)}$, and choosing the step $\alpha^{(k)}$ by

$$\alpha = \min\left(1, \min_{\substack{i:i\notin\mathcal{A}, \\ a_i^T s^{(k)}<0}} \frac{b_i - a_i^T x^{(k)}}{a_i^T s^{(k)}}\right) \tag{8.3}$$

so that $x^{(k+1)} = x^{(k)} + \alpha^{(k)}s^{(k)}$. If $\alpha^{(k)} < 1$, then a new constraint becomes active, defined by the index (p say) which achieves the minimum and this index is added to the active set $\mathcal{A}$. If there are more than one constraint with the minimum value, all of them are added to the active set.

If $x^{(k)}$ (that is, $\delta^k = 0$) solves the current EP, then it is possible to compute multipliers ($\lambda^{(k)}$ say) for the active constraints as described in the two methods given below. The vectors $x^{(k)}$ and $\lambda^{(k)}$ then satisfy all the first order optimality conditions for the inequality constraint problem except possibly that $\lambda_i \geq 0, i \in I$. Thus, a test is made to determine whether $\lambda_i^{(k)} \geq 0 \ \forall i \in \mathcal{A} \cap I$; if so, then the first order optimality conditions are satisfied and these are sufficient to ensure a global solution since $q(x)$ is convex. Otherwise there exists an index, r say, $r \in \mathcal{A} \cap I$ such that $\lambda_r^{(k)} < 0$. In this case, it is possible to reduce $q(x)$ by allowing the constraint r to become inactive. Thus r is removed from $\mathcal{A}$ and the algorithm continues as before by solving the resulting EP. If there is more than one index r for which $\lambda_r^{(k)} < 0$ then r is selected to solve

$$\min_{i\in\mathcal{A}\cap I} \lambda_i^{(k)}. \tag{8.4}$$

In the case of a tie, all the constraints that correspond to the minimum of the negative λa are made inactive.

Algorithms to solve an EP

The aim is to find a solution x^* to the equality constrained problem

$$\begin{aligned} &\underset{x}{\text{minimize}}\, q(x) \equiv \frac{1}{2}x^T G x + q^T x, \\ &\text{subject to } A^T x = b. \end{aligned} \tag{8.5}$$

It is assumed that there are $m \leq n$ constraints, so that $b \in \mathbf{R}^m$, and A is $n \times m$ with the column vectors a_i, $i \in E$. It is assumed that A has rank m.

Method 1

One way of solving the EP is to use the constraints to eliminate variables. If the partitions

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \qquad A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \qquad g = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}, \qquad G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \tag{8.6}$$

are defined, where $x_1 \in \mathbf{R}^m$ and $x_2 \in \mathbf{R}^{n-m}$, etc., and rank $A_1 = m$, then the constraints become $A_1^T x_1 + A_2^T x_2 = b$ and are readily solved (by Gaussian Elimination) for x_1 in terms of x_2:

$$x_1 = A_1^{-T}(b - A_2^T x_2).$$

Substituting into $q(x)$ gives the problem: minimize $\psi(x_2)$, $x_2 \in \mathbf{R}^{n-m}$ where $\psi(x_2)$ is the quadratic function

$$\begin{aligned} \psi(x_2) = \frac{1}{2}x_2^T &\left(G_{22} - G_{21}A_1^{-T}A_2^T - A_2A_1^{-1}G_{12} + A_2A_1^{-1}G_{11}A_1^{-1}A_2^T\right)x_2 \\ &+ x_2^T\left(G_{21} - A_2A_1^{-1}G_{11}\right)A_1^{-T}b + \frac{1}{2}b^TA_1^{-1}G_{11}A_1^{-T}b \\ &+ x_2^T\left(g_2 - A_2A_1^{-1}g_1\right) + g_1^TA_1^{-T}b. \end{aligned} \tag{8.7}$$

The Lagrange multiplier vector λ^* is defined by $g^* = \nabla q(x^*) = A\lambda^*$ and can be calculated by solving $g^* = \nabla q(x^*)$ the first partition $g_1^* = A_1\lambda^*$. By the definition of $q(x)$, $g^* = g + Gx^*$, so an explicit expression for λ^* is

$$\lambda^* = A_1^{-1}\left(g_1 + G_{11}x_1^* + G_{12}x_2^*\right). \tag{8.8}$$

Method 2

A generalized elimination method is possible in which essentially a linear transformation of variables is made initially. Let S and Z be $n \times m$ and $n \times (n-m)$ matrices respectively such that $[S : Z]$ is nonsingular, and in addition let $A^T S = I$ and $A^T Z = 0$. S^T can be regarded as a left generalized inverse for A so that a solution of $A^T x = b$ is given by $x = Sb$. However this solution is non-unique in general and other feasible points are given by $x = Sb + \delta$ where δ is in the null space of A^t.

$$\{\delta \mid A^T \delta = 0\},$$

which has dimension $n - m$. The columns $z_1, z_2, \ldots, z_{n-m}$ of Z are a basis for this null space. That is to say, any feasible correction δ can be written as

$$\delta = Zy = \sum_{i=1}^{n-m} z_i y_i, \tag{8.9}$$

where $y_1, y_2, \ldots, y_{n-m}$ are the components in each reduced coordinate direction. Thus, any feasible point x can be written as

$$x = Sb + Zy.$$

Substituting into $q(x)$ gives

$$\psi(y) = \frac{1}{2} y^T Z^T G Z y + (g + GSb)^T Z y + \frac{1}{2}(g + GSb)^T Sb. \tag{8.10}$$

Since $Z^T G Z$ is positive definite a unique minimizer y^* exists which solves the linear system

$$(Z^T G Z) y = -Z^T (g + GSb).$$

Now x^* is obtained by substitution,

$$x^* = Sb - Z(Z^T G Z)^{-1} Z^T (g + GSb)$$

$g^* = A\lambda^*$ gives $\lambda^* = S^T g^*$ which can be used to calculate Lagrange multipliers,

$$\lambda^* = S^T(g + Gx^*) = \left(S^T - S^T G Z (Z^T G Z)^{-1} Z^T\right) g + S^T \left(G - G Z (Z^T G Z)^{-1} Z^T G\right) Sb.$$

Depending on the choice of S and Z, a number of methods exist, but one numerically stable way uses the QR factorization of the matrix A.

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = [Q1 \quad Q2] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R, \tag{8.11}$$

where Q is $n \times n$ and orthogonal, R is $m \times m$ and upper triangular and Q_1 and Q_2 are $n \times m$ and $n \times (n-m)$, respectively. S and Z are chosen as follows:

$$S = A^{+T} = Q_1 R^{-T}, \quad Z = Q_2, \tag{8.12}$$

where A^+ denotes the Moore Penrose inverse of A.

Method 2 was used in the implementation with S and Z chosen using the QR factorization of the matrix A.

8.2 QPSOL.

QPSOL [16] is a FORTRAN package for solving quadratic programming problems. It is designed to solve a QP of the following form:

$$\begin{aligned} &\underset{x}{\text{minimize}} \; f(x) \\ &\text{subject to } l \leq \begin{Bmatrix} x \\ Ax \end{Bmatrix} \leq u, \end{aligned} \tag{8.13}$$

where A is $m_L \times n$ (m_L may be zero) and $f(x)$ is either a linear or a quadratic objective function. It can be used to find a feasible point for a set of constraints, the corresponding objective function is then set to zero.

QPSOL is based on a type of active set strategy called an inertia-controlling method, due to Gill and Murray [16]. Briefly, the algorithm used in QPSOL has two phases, a feasibility phase and an optimality phase.

The feasibility phase deals with finding an initial feasible point and the optimality phase concentrates on minimizing the quadratic objective function within the feasible region. The two phases are performed by the same routines, simply changing the function to be minimized from the sum of the infeasibilities to the quadratic objective function.

8.3 MINOS.

MINOS [23] is an optimization package designed to solve problems in

- linear programming,
- unconstrained optimization,
- linearly constrained optimization, and
- nonlinearly constrained optimization.

Thus, the optimization problem solved can be expressed as:

$$\begin{aligned} &\underset{x,y}{\text{minimize}}\, F(x) + c^T x + d^T y \\ &\text{subject to } f(x) + A_1 y = b_1, \\ &\qquad A_2 x + A_3 y = b_2, \\ &\qquad\qquad l \le \begin{pmatrix} x \\ y \end{pmatrix} \le u, \end{aligned} \tag{8.14}$$

where the vectors c, d, b_1, b_2, l, u and the matrices A_1, A_2, A_3 are constant, $F(x)$ is a smooth scalar function, and $f(x)$ is a vector of smooth functions. The n_1 components of x are called the nonlinear variables, and the n_2 components of y are the linear variables. The first set of equations are the nonlinear constraints, the second set the linear constraints, and the third set the upper and lower bounds on all the variables. There is a provision for introducing upper and lower bounds on the constraints as well. Such constraints would be of the form

$$l_1 \le f(x) + A_1 \le u_1,$$

$$l_2 \le A_2 x + A_3 y \le u_2,$$

but are actually specified in terms of a right-hand side b_i and a range $u_i - l_i$.

The algorithms that MINOS is based on are

- the simplex method, (Dantzig [9]),
- a quasi-Newton method, (Davidon [10]),
- the reduced-gradient method, (Wolfe [46]), and
- a projected Lagrangian method (Robinson [28]; Rosen and Kreuser [29]).

9. HISTORICAL DEVELOPMENT OF PRESENT ALGORITHM.

The original algorithm presented by Sobieski [33] was used without any modifications initially on a 2×2 test problem. This chapter gives a description of the main problems encountered at the beginning and the modifications that were made to the original algorithm to resolve these problems, with the results tabulated for every stage. The tables are given together at the end for ease of comparison.

Also, a brief description of some of the later modifications attempted that were not successful is given in the end.

9.1. The switch coefficient s.

A simple 2×2 quadratic programming problem (5.1) was used in the beginning to better understand any problems encountered with either the algorithm or the implementation. The results of testing the original algorithm on this example for 5 different starting points and βs are tabulated in Table I in Chapter 5. There the t-coefficients are bounded in magnitude by 1.0. The original algorithm with the t-coefficients left unbounded (see Section 9.3) led to Table VIII. On examining the algorithm [33] closely, one major problem was detected. The s-coefficient, which is a switch, turns one of the r or the t-coefficients on, depending on whether the corresponding constraint is violated or satisfied, respectively. But once the s-coefficient is set to zero, it stays at zero until the entire procedure terminates. Consider the following cumulative constraint of the pth subsystem present in the ith subsystem:

$$C_i^p(X^i, Y^i) \leq \hat{K}_p(X_0^i, Y_0^i)\, s^p(1 - r_i^p) + (1 - s^p)t_i^p. \tag{9.1}$$

If the s-coefficient s^p activates t_i^p, then r_i^p can never be activated again, even in the event of a violation. This is not right, as the purpose of the r-coefficient is to help in reducing constraint violation.

Two different solutions were suggested to this problem. One was to switch the s-coefficient on and off depending on whether the constraint is currently violated or not. Thus, at the end of every outer iteration, the s-coefficients are updated after checking to see if the corresponding constraints are violated or not.

The original algorithm with this change was tried out on the 2×2 example (with the t-coefficients left unbounded, see Section 9.3) and the results are given in Table IX.

The other solution was to remove the s-coefficient from the r term. Thus, the r-coefficient is always active, both in the case of a violation and satisfaction. Equation (9.1) would then become

$$C_i^p(X^i, Y^i) \leq \hat{K}_p(X_0^i, Y_0^i)\,(1 - r_i^p) + (1 - s^p)t_i^p. \tag{9.2}$$

This modification was also tried on the 2×2 example and the results (no bounds on the t-coefficients, see Section 9.3) are given in Table X. Since the results corresponding to the first solution were better, this was used for further testing.

9.2. An infeasible subproblem.

As is evident from the Tables VIII, IX, and X corresponding to the above mentioned changes, when an infeasible subproblem was encountered, the original algorithm had no way of dealing with it. The procedure had to terminate abnormally at this point. Thus one of the first few modifications was to deal with infeasibility in a subproblem.

This is done by introducing a new variable ω in the corresponding subsystem. The variable ω is added to each of the constraints and a large multiple of this variable is added to the objective function. The infeasible subsystem is now solved once again with ω. Precisely, the modified subsystem is:

$$\begin{aligned} &\min_{X^i} \hat{\Theta}(X^i) + M\omega \\ \text{subject to} \quad &\hat{g}^i(X^i, Y^i) - \omega \leq s^i \max\{\hat{g}^i(X_0^i, Y_0^i), 0\}(1 - r_i^i) + (1 - s^i)t_i^i, \\ &C_i^p(X^i, Y^i) - \omega \leq \hat{K}_p(X_0^i, Y_0^i)\, s^p(1 - r_i^p) + (1 - s^p)t_i^p, \\ &\qquad\qquad\qquad\qquad\qquad\qquad p = 1, \ldots, i-1, i+1, \ldots, N, \end{aligned} \tag{9.3}$$

where M is a large positive number.

The coefficients of the variables in the objective function of the COP involve calculation of the sensitivity derivatives according to the method given in Appendix A. The Lagrange multipliers obtained from solving the individual subproblems are used to obtain these derivatives.

When ω is introduced in a subsystem to deal with infeasibility, the corresponding Lagrange multipliers are also affected. A thought as to whether these Lagrange multipliers can be rightfully allowed to affect the COP led to two different versions. In one version the COP is performed at the end of every outer iteration, and in the other whenever there is an infeasibility in any subproblem and ω is used to recover from the infeasibility, the COP is not performed in the corresponding outer iteration. This leads to the two Tables XI and XII. Both of them correspond to the s-coefficient being updated at the end of every outer iteration. Table XII corresponds to the COP being included and Table XI corresponds to the COP not being present in the case of an infeasibility. Both these Tables correspond to the t being unbounded.

9.3. Bounds on the t-coefficients.

The t-coefficients allow a violation of a constraint in one subsystem, provided it is compensated for by an oversatisfaction in another subsystem. Initially tests were conducted with these t-coefficients left unbounded, and the results are given in Tables XI and XII. This implies that the t-coefficient can be any large value, which means that a constraint can be violated by a large amount in a particular subsystem. This is not right. Therefore bounds were introduced for the t-coefficients, which were a feature of the original algorithm also.

Different ways of handling these bounds were tried. One approach was to fix the bound at a certain value initially, and then have this bound decrease by a certain factor after a specified number of iterations.

Tables XI and XII (Section 9.4) had the t-coefficients unbounded. This modified version but with the t being bounded in magnitude by 1.0 initially and having these bounds reduced by a factor of 0.8 at the end of every outer iteration, starting from the first, yielded Tables XIII and XIV. Having the bound always fixed at 1.0 gives Table XV.

The bound on the t needs to be reduced in some fashion, especially when close to the solution. Thus, other ways of reducing these bounds were tried. One was to reduce the bound by a logarithmic function (see Chapter 6). The geometric update yielded the best results, although the logarithmic update also was good enough.

9.4. A description of the different versions.

The following list gives the characteristics of the different versions of the algorithm that are tabulated at the end.

1. The original algorithm with the s-coefficient set to zero when the constraint is satisfied and held at zero thereafter. Exits with a message in the case of an infeasibility. The value of the t-coefficient is unbounded.

2. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. Exits with a message in the case of an infeasibility. The value of the t-coefficient is unbounded.

3. The s-coefficient is removed from the r term in each of the constraints in the subsystem. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. Exits with a message in the case of an infeasibility. The value of the t-coefficient is unbounded.

4. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. A new variable ω is introduced in every constraint to deal with infeasibility. A large multiple of ω is added to the objective function. There is no Coordination Optimization Phase in the case of an infeasibility. The value of the t-coefficient is unbounded.

5. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. A new variable ω is introduced in every constraint to deal with infeasibility. A large multiple of ω is added to the objective function. There is a Coordination Optimization Phase at the end of every outer iteration. The value of the t-coefficient is unbounded.

6. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is

satisfied. A new variable ω is introduced in every constraint to deal with infeasibility. A large multiple of ω is added to the objective function. There is no Coordination Optimization Phase in the case of an infeasibility. The value of the t-coefficient is bounded initially between -1 and 1 and these bounds are multipled by a factor of 0.8 at the end of every outer iteration starting from the first.

7. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. A new variable ω is introduced in every constraint to deal with infeasibility. A large multiple of ω is added to the objective function. There is a Coordination Optimization Phase at the end of every outer iteration. The value of the t-coefficient is bounded initially between -1 and 1 and these bounds are multipled by a factor of 0.8 at the end of every outer iteration starting from the first.

8. At the end of every outer iteration the s-coefficient corresponding to each of the cumulative constraints is set to 1 if the constraint is not satisfied and to 0 if it is satisfied. A new variable ω is introduced in every constraint to deal with infeasibility. A large multiple of ω is added to the objective function. There is no Coordination Optimization Phase. The value of the t-coefficient is bounded initially between -1 and 1 and these bounds are held fixed.

KEY TO THE TABLES

1. C–Converges to the right value.
2. NC–No Convergence.
3. WC–Converges to a wrong value.
4. AC–Converges to approximately the right value.
5. CD–Reaches the correct value and then diverges.
6. IF–Problem is infeasible.
7. ACD–Reaches approximately the right value and then. diverges
8. ACO–Reaches approximately the right value and then oscillates.
9. O–Oscillates.

TABLE VIII
Original Algorithm
t unbounded.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	C 1	C 1	C 1	C 1	C 1	(0.0, 2.0)
0.1	O 30	IF 1	WC 1	WC 1	IF 1	(0.198, 1.98)
0.3	O 30	IF 1	WC 1	WC 1	IF 1	(0.55, 1.835)
0.5	O 30	IF 1	WC 1	AC 1	IF 2	(0.8, 1.6)
1.0	O 30	WC 2	WC 1	O 30	O 30	(1.0, 1.0)

TABLE IX
s updated and t unbounded.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	C 1	C 1	C 1	C 1	C 1	(0.0, 2.0)
0.1	AC 2	IF 1	WC 1	WC 1	IF 1	(0.198, 1.98)
0.3	AC 2	IF 1	WC 1	WC 1	IF 1	(0.55, 1.835)
0.5	AC 2	IF 1	WC 1	AC 1	WC 2	(0.8, 1.6)
1.0	C 2	WC 2	WC 1	WC 2	WC 2	(1.0, 1.0)

TABLE X
s removed and t unbounded.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	C 1	C 1	C 1	C 1	C 1	(0.0, 2.0)
0.1	WC 2	IF 1	WC 1	WC 1	IF 1	(0.198, 1.98)
0.3	WC 2	IF 1	WC 1	WC 1	IF 1	(0.55, 1.835)
0.5	WC 2	IF 1	WC 1	AC 1	IF 1	(0.8, 1.6)
1.0	WC 2	WC 2	WC 1	WC 2	WC 2	(1.0, 1.0)

TABLE XI
s updated, ω used, no COP
t unbounded.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	C 1	C 1	C 1	C 1	C 1	(0.0, 2.0)
0.1	AC 2	WC 2	WC 1	WC 1	WC 2	(0.198, 1.98)
0.3	AC 2	NC 20	WC 1	WC 1	WC 2	(0.55, 1.835)
0.5	AC 2	NC 20	WC 1	AC 1	WC 2	(0.8, 1.6)
1.0	C 2	WC 2	WC 1	WC 2	WC 2	(1.0, 1.0)

TABLE XII
s updated, ω used, COP
t unbounded.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	C 1	C 1	C 1	C 1	C 1	(0.0, 2.0)
0.1	AC 2	WC 2	WC 1	WC 1	WC 2	(0.198, 1.98)
0.3	AC 2	NC 20	WC 1	WC 1	WC 2	(0.55, 1.835)
0.5	AC 2	WC 3	WC 1	AC 1	WC 2	(0.8, 1.6)
1.0	C 2	WC 2	WC 2	WC 2	WC 2	(1.0, 1.0)

TABLE XIII
s updated, ω used, no COP
t bound at 1 and 0.8 update.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	CD 1	CD 3	CD 3	CD 3	CD 1	(0.0, 2.0)
0.1	ACD 3	WC 2	NC 100	NC 100	WC 2	(0.198, 1.98)
0.3	NC 100	NC 100	NC 100	NC 100	WC 2	(0.55, 1.835)
0.5	AC 16	NC 20	AC 10	AC 10	AC 10	(0.8, 1.6)
1.0	C 2	AC 13	AC 15	C 15	AC 17	(1.0, 1.0)

TABLE XIV
s updated, ω used, COP
t bound at 1 and 0.8 update.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	CD 4	CD 3	CD 3	CD 3	CD 4	(0.0, 2.0)
0.1	ACD 2	NC 20	NC 20	NC 20	NC 20	(0.198, 1.98)
0.3	NC 20	NC 20	NC 20	NC 20	CD 18	(0.55, 1.835)
0.5	AC 16	AC 16	AC 10	AC 10	AC 10	(0.8, 1.6)
1.0	C 2	AC 13	AC 15	AC 15	AC 17	(1.0, 1.0)

TABLE XV
s updated, ω used, no COP
t bound at 1 and bound is fixed.

*	(2,3)	(4,-1)	(1,-1)	(0.8,1.5)	(10,3)	OPT
0.0	CD 4	CD 3	CD 3	CD 3	CO 1	(0.0, 2.0)
0.1	ACD 2	WC 2	WC 4	WC 4	WC 2	(0.198, 1.98)
0.3	WC 3	WC 6	WC 5	WC 4	WC 2	(0.55, 1.835)
0.5	CO 20	WC 6	ACD 2	O 20	O 20	(0.8, 1.6)
1.0	C 2	O 20	O 20	O 20	WC 2	(1.0, 1.0)

9.5. Some later modifications that were tested.

The current version of the algorithm evolved from the original one after a number of modifications were made along the way. Some of them worked and some of them did not work as they had been expected to and so were dropped along the way.

This section gives some of the later modifications attempted that were not successful.

1. In the last phase of the work, before it was discovered that there could be a problem with the sensitivity information used in the COP (see Chapter 7), a number of attempts were made to arrive at the right set of t-coefficients at every iteration. It was observed that a particular t-coefficient could jump from one extreme of the t range to the other over two successive iterations. Move limits were introduced on the t-coefficients to prevent this jump, but this did not help in obtaining the right set of t-coefficients.
2. The algorithm that works for the 2×2 and 3×3 examples does not work well for the 6×6 case. An attempt was made to check if with a small bound on the t-coefficients there was convergence to the solution after a certain number of iterations. If the process converged to a wrong point, then a cold start was performed at this point (i.e., the whole process is restarted with this wrong point as the starting point). This did not help in obtaining any better results, which should be so, as this version of the algorithm diverges away from the solution even when the correct solution is used as the starting point.

10. Conclusions.

Despite the success reported by [7] with the original algorithm of Sobieski [32], we have required major modifications even for a 2×2 quadratic program. For larger QPs with weak subsystem coupling, the modified algorithm described in Chapter 7 works reasonably well. For larger QPs with strong coupling, the sensitivities obtained from the parallel subsystem optimizations were too unreliable for the COP to produce rational changes in the r- and t-coefficients. One possibility for correcting this problem may be to compute subsystem sensitivities with a global (and hence serial) step, as advocated in [27].

[1] F. F. ABDI, H. IDE, V. J. SHANKAR, AND J. SOBIESZCZANSKI-SOBIESKI, *Optimization for nonlinear aeroelastic tailoring criteria*, Int. Coun. Aero. Sci., 16th Congress, Aug., 1988, Jerusalem.

[2] J.-F. M. BARTHELEMY AND J. SOBIESZCZANSKI-SOBIESKI, *Optimum sensitivity derivatives of objective functions in nonlinear programming*, AIAA J., 21 (6) (1983), pp. 913–915.

[3] J.-F. M. BARTHELEMY AND M. F. RILEY, *Improved multi-level optimization approach for the design of complex engineering systems*, AIAA J., 26(3) (1988), pp. 353–360.

[4] J.-F. M. BARTHELEMY, *Engineering applications of multilevel optimization*, Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, Sept., 1988, Hampton, VA.

[5] D. P. BERTSEKAS, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 1982.

[6] C. L. BLOEBAUM AND P. HAJELA, *Heuristic Decomposition for Non-Hierarchic Systems*, AIAA/ASME/ASCE/AHS 32nd Structures, Structural Dynamics, and Materials Conference, Baltimore, MD, April, 1991, pp. 344–353.

[7] C. L. BLOEBAUM, P. HAJELA AND J. SOBIESZCZANSKI-SOBIESKI, *Non-Hierarchic System Decomposition in Structural Optimization*, Third NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, Sept., 1990, San Francisco, CA.

[8] A. C. DAMOTA, *Analysis of parallelism for Non linear programming*, M.S. Thesis, Instituto de Pesquisas Espaciais, Sao Jose dos Campos.

[9] G. B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

[10] W. C. DAVIDON, *Variable metric methods for minimization*, A. E. C. Res. and Develop. Report ANL-5990, Argonne National Laboratory, Argonne, IL, 1959.

[11] G. Di Pillo, F. Facchinei and L. Grippo, *A RQP algorithm using a differentiable exact penalty function for inequality constrained problems*, Technical Report 03.89, Dept. of Information and Systems, Rome, Italy, 1989.

[12] R. Fletcher, *Practical methods of optimization Vol. 2*, John Wiley and Sons, New York, 1980.

[13] J. C. Gilbert, *Maintaining the positive definiteness of the matrices in reduced secant methods for equality constrained optimization*, IIASA working paper WP-87-123, IIASA, Laxenburg, Austria, 1987.

[14] P. E. Gill, W. Murray, M. A. Saunders and W. H. Wright, *Inertia-controlling methods for quadratic programming*, SIAM Review, 33 (1991), pp. 1–36.

[15] P. E. Gill, W. Murray, M. A. Saunders and W. H. Wright, *Procedures for optimization problems with a mixture of bounds and general constraints*, ACM Transactions on Mathematical Software, 10 (1984), pp. 282–298.

[16] P. E. Gill, W. Murray and M. A. Saunders, *QPSOL*, Department of Operations Research, Stanford University, Stanford, CA, 1984.

[17] G. Kreisselmeier and R. Steinhauser, *Systematic control design by optimizing a vector performance index*, Proc. IFAC Symp. on Computer Aided Design of Control Systems, Zurich, Switzerland, (1979), pp. 113–117.

[18] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Technical Report NAM 03, Dept. of EE and CS, Northwestern Univ., Evanston, IL, 1988.

[19] M. Mckema and S. A. Zenios, *The optimal implementation of a quadratic transportation algorithm*, Proc. of the 4th SIAM Conference on Parallel Processing for Scientific Comput., Chicago, IL, Dec, 1989.

[20] B. Mendelson and I. Koren, *Estimating the potential parallelism and pipelining of algorithms for data flow machines*, J. Parallel and Distributed Comput. 14, Jan. (1992), pp. 15–28.

[21] B. A. MURTAGH AND M. A. SAUNDERS, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Prog. Study 16 (1982), pp. 84–117.

[22] B. A. MURTAGH AND M. A. SAUNDERS, *Large scale linearly constrained optimization*, Math. Prog. 14 (1978), pp. 41–72.

[23] B. A. MURTAGH AND M. A. SAUNDERS, *MINOS 5.0 User's Guide*, Report SOL 83-20, Department of Operations Research, Stanford University, Stanford, CA, 1983.

[24] J. PAN, AND A. R. DIAZ, *Some results in Optimization of Non-Hierarchic Systems*, Advances in Design Automation, ASME Publication DE-Vol.19-2, 1989.

[25] E. R. PANIER AND A. L. TITS, *On feasibility, descent and superlinear convergence in inequality constrained optimization*, Technical Research Report 89-27, Systems Research Center, Univ. of Maryland, College Park, MD, 1989.

[26] A. R. PARKINSON, R. J. BALLING, A. WU AND J. C. FREE, *A general strategy for decomposing large design problems based on optimization and statistical test plans*, International Conference on Engineering Design, ICED 87, Boston, MA, Aug., 1987, pp. 162.

[27] J. E. RENAUD AND G. A. GABRIELE, *Sequential global approximation in non-hierarchic system decomposition and optimization*, Advances in Design Automation, 1 (1991), pp. 191–200.

[28] S. M. ROBINSON, *A quadratically convergent algorithm for general nonlinear programming problems*, Math. Prog, 3 pp. 145–156, 1972.

[29] J. B. ROSEN AND J. KREUSER, *A gradient projection algorithm for nonlinear constraints*, Numerical Methods for Non-Linear optimization (F. A. Lootsma, ed.) pp 297–300, Academic Press, London and NY 1972.

[30] Y. S. SHIN, R. T. HAFTKA, L. T. WATSON, AND R. H. PLAUT, *Tracing structural optima as a function of available resources by a homotopy method*, Comput. Methods Appl. Mech. Engrg., 70 (1988), pp. 151–164.

[31] Y. S. SHIN, R. T. HAFTKA, L. T. WATSON, AND R. H. PLAUT, *Design of laminated plates for maximum buckling load*, J. Composite Materials, 23 (1989), pp. 348–369.

[32] J. SOBIESZCZANSKI-SOBIESKI, *Optimization by decomposition: a step from hierarchic to nonhierarchic systems*, Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, Sept., 1988, Hampton, VA.

[33] J. SOBIESZCZANSKI-SOBIESKI, *A linear decomposition method for large optimization problems*, NASA TM 83248, Feb., 1982.

[34] J. SOBIESZCZANSKI-SOBIESKI, B. B. JAMES, AND M. F. RILEY, *Structural sizing by generalized, multilevel optimization*, AIAA J., 25(1) (1987), p. 139-145.

[35] J. SOBIESZCZANSKI-SOBIESKI, *On the sensitivity of complex, internally coupled systems*, AIAA/ASME/ASCE/AHS 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA, April, 1988.

[36] J. SOBIESZCZANSKI-SOBIESKI, J.-M. F. BARTHELEMY, AND K. M. RILEY, *Sensitivity of optimum solutions to problem parameters*, AIAA J., 20 (1982), pp. 1291–1299.

[37] J. SOBIESZCZANSKI-SOBIESKI, *Two alternative ways for solving the coordination problem in multilevel optimization*, NTIS HC/MF A03.

[38] O. O. STORAASLI AND E. A. CARMORA, *Parallel methods on large scale structural analysis and physics applications*, Computing Systems in Engineering, 2(2-3), (1991), pp.199.

[39] G. N. VANDERPLAATS, Y. J. YANG, AND D. S. KIM, *An efficient multilevel optimization method for engineering design*, AIAA/ASME/ASCE/AHS 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA, April, 1988, pp. 125–132.

[40] G. N. VANDERPLAATS, AND H. MIURA, *Computational trends in large scale engineering optimization*, Applications of supercomputers in engineering: Fluid flow and stress analysis applications; Proceedings of the first international conference, Southampton, England, 2, Sep, (1989), pp. 269–283, Elsevier/Computational Mechanics Publications.

[41] G. N. VANDERPLAATS, AND H. MIURA, *Experiences in large scale structural design optimization*, Applications of supercomputers in engineering: Fluid flow and stress

analysis applications; Proceedings of the first international conference, Southampton, England, 2, Sep, (1989), pp. 269–283, Elsevier/Computational Mechanics Publications.

[42] G. VASUDEVAN, L. T. WATSON, AND F. H. LUTZE, *A homotopy approach for solving constrained optimization problems*, Tech. Rep. 88–50, Dept. of Computer Sci., VPI&SU, Blacksburg, VA, 1988.

[43] L. T. WATSON, R. T. HAFTKA, F. H. LUTZE, R. H. PLAUT, AND P. Y. SHIN, *The application of globally convergent homotopy methods to nonlinear optimization*, Advances in Numerical Partial Differential Equations and Optimization, S. Gómez, J. P. Hennart, and R. A. Tapia (eds.), SIAM, Philadelphia, PA, 1991, pp. 284–298.

[44] L. T. WATSON, S. C. BILLUPS, AND A. P. MORGAN, *HOMPACK: A suite of codes for globally convergent homotopy algorithms*, Tech. Rep. 85–34, Dept. of Industrial and Operations Eng., Univ. of Michigan, Ann Arbor, MI, 1985, and ACM Trans. Math. Software, 13 (1987), pp. 281–310.

[45] L. T. WATSON AND W. H. YANG, *Optimal design by a homotopy method*, Applicable Anal., 10 (1980), pp. 275–284.

[46] P. WOLFE, *The reduced-gradient method*, unpublished manuscript, RAND Corporation, 1962.

[47] G. A. WRENN AND A. R. DOVI, *Multilevel decomposition approach to the preliminary sizing of a transport aircraft wing*, AIAA/ASME/ASCE/AHS 29th Structures, Structural Dynamics, and Materials Conference, Williamsburg, VA, April, 1988.

[48] B. C. WU, AND S. AZARM, *An approach for optimization based design of non-hierarchic systems*, Third Air Force/NASA Symposium on Recent Advances in Multidisciplinary Optimization, San Francisco, CA, Sep. 24-26, 1990.

Appendix A.

Initialization of the r-coefficients.

The coefficients may be initialized on the basis of sensitivity information so as to assign a greater responsibility for a cumulative constraint satisfaction (of the ith subsystem say) to those subsystems that have a greater influence on that constraint. Let

$$(K_{pk})_i \equiv \frac{\partial K_p}{\partial X_i^k}(x_0, y_0).$$

Since $1 \le p \le N$, $1 \le k \le N$, and $1 \le i \le n_k$, there are Nn_k such partial derivatives $(K_{pk})_i$, for every k. Now define

$$a^{pk} \equiv \max_{1 \le i \le n_k} \left|(K_{pk})_i\right|, \qquad 1 \le p \le N, \quad 1 \le k \le N,$$

which measures the influence of the kth subsystem's variables X^k on the pth subsystem's constraints, as represented by K_p. Normalizing these N^2 influence coefficients gives the r-coefficients

$$r_k^p = \frac{a^{pk}}{\sum_{j=1}^{N} a^{pj}}, \qquad 1 \le p \le N, \quad 1 \le k \le N.$$

Optimum Sensitivity Analysis.

Let z denote either of r_i^p or t_i^p, and define the modified constraint functions

$$\tilde{g}^i(X^i, Y^i) = \hat{g}^i(X^i, Y^i) - \left[s^i \max\{\hat{g}^i(X_0^i, Y_0^i), 0\}(1 - r_i^i) + (1 - s^i)t_i^i\right],$$
$$\tilde{C}_i^p(X^i, Y^i) = C_i^p(X^i, Y^i) - \left[\hat{K}_p(X_0^i, Y_0^i)\, s^p(1 - r_i^p) + (1 - s^p)t_i^p\right],$$
$$i = 1, \ldots, N, \quad p = 1, \ldots, i-1, i+1, \ldots, N.$$

Let $\nabla_i = \left(\frac{\partial}{\partial X_1^i}, \ldots, \frac{\partial}{\partial X_{n_i}^i}\right)$,

$$G^i = \begin{pmatrix} \tilde{g}^i \\ \tilde{C}_i^1 \\ \vdots \\ \tilde{C}_i^{i-1} \\ \tilde{C}_i^{i+1} \\ \vdots \\ \tilde{C}_i^N \end{pmatrix},$$

and G_A^i denote the subvector of G^i corresponding to the active constraints at the current point. It is assumed that the dimension of G_A^i is less than or equal to n_i, and that the Jacobian matrix $\nabla_i G_A^i$ has full rank. Then the sensitivities of the minimum of Θ with respect to the constraints $\tilde{g}^i \leq 0$, $\tilde{C}_i^p \leq 0$ are given by the Lagrange multipliers

$$\lambda = -\left[(\nabla_i G_A^i)(\nabla_i G_A^i)^t\right]^{-1} (\nabla_i G_A^i)(\nabla_i \Theta)^t,$$

where everything is evaluated at the current point—the result of the Nth subsystem optimization. Now from this the sensitivities of the minimum of Θ with respect to the r_i^p and t_i^p are given by

$$\frac{\partial \Theta}{\partial z} = \lambda^t \frac{\partial G_A^i}{\partial z}.$$

Observe that from the form of G^i, the partials $\partial G_A^i / \partial z$ are trivial to compute. λ would not be computed explicitly from the projection operator as described above, but rather from a QR factorization of $(\nabla_i G_A^i)^t$, as described in Fletcher [12].

Appendix B.

```
c*************************************************************************
c     THIS ROUTINE SOLVES AN OPTIMIZATION PROBLEM WITH A QUADRATIC
c     OBJECTIVE FUNCTION AND LINEAR CONSTRAINTS, BY DIVIDING THE
c     SYSTEM INTO k DIFFERENT SUBSYSTEMS AND SOLVING EACH INDEPENDENTLY.
c     THERE IS A FINAL COORDINATION PHASE THAT COORDINATES THE RESULTS
c     OBTAINED FROM THE DIFFERENT SUBSYSTEMS. THE PROCESS IS CARRIED
c     ON ITERATIVELY UNTIL THE DESIRED CONVERGENCE CRITERION IS MET.
c*************************************************************************
c
      subroutine nhdec(maxsub,maxvar,maxcon,amat,cmat,alph,beta,
     &                 d,x,delx,tlim,rho,CONTOL,sum,
     &                 k,n,np,itfix,itval)
c     maxsub GIVES THE MAXIMUM NUMBER OF SUBSYSTEMS
c     maxvar GIVES THE MAXIMUM NUMBER OF VARIABLES IN ANY SUBSYSTEM
c     maxvar SHOULD BE THE MAX NO OF VAR + 1 TO ACCOUNT FOR OMEGA,
c           A VARIABLE INTRODUCED TO DEAL WITH INFEASIBILITY
c     maxcon GIVES THE MAXIMUM NUMBER OF CONSTRAINTS IN ANY SUBSYSTEM
c     amat THE OBJECTVE FUNCTION MATRICES Aij ARE GIVEN ROW BY ROW USING
c          THE ARRAY amat
c     cmat THE CONSTRAINT MATRICES ARE GIVEN USING THE ARRAY cmat
c     alph AND beta GIVE THE ALPHA AND BETA COEFFICIENTS
c     d GIVES THE RHS OF THE CONSTRAINTS
c     x GIVES A STARTING POINT ON INPUT AND THE FINAL DESIGN VECTOR
c       VALUE ON OUTPUT
c     delx IS THE MOVE LIMIT ON x
c     tlim IS THE INITIAL BOUND ON t
c     rho IS THE VALUE OF RHO
c     CONTOL IS THE DESIRED CONVERGENCE VALUE
c     sum GIVES THE VALUE OF THE OBJECTIVE FUNCTION ON OUTPUT
c
c     k GIVES THE NUMBER OF SUBSYSTEMS,
c     n AND np GIVE THE NUMBER OF VARIABLES AND
c             CONSTRAINTS IN EACH SUBSYSTEM RESPECTIVELY
c     itfix IS THE NUMBER OF ITERATIONS THAT THE LIMIT ON t IS HELD FIXED
c          AT tlim BEFORE BEING REDUCED
c     itval IS THE LIMIT ON THE TOTAL NUMBER OF OUTER ITERATIONS
c
      integer maxsub,maxvar,maxcon,k,n(maxsub),np(maxsub),itfix,itval
      double precision amat(maxsub**2,maxvar,maxvar),
     &                 cmat(maxsub**2,maxcon,maxvar),
     &                 d(maxsub,maxcon),x(maxsub,maxvar),
     &                 alph(maxsub,maxsub),beta(maxsub,maxsub),
     &                 delx,rho,tlim,CONTOL,sum
      double precision prevx(maxsub,maxvar),kpo(maxsub),
     &                 apk(maxsub,maxsub),rhox(maxsub,maxvar),
     &                 newx(maxsub,maxvar),mlam(maxsub,maxcon),
     &                 rpk(maxsub,maxsub),prpk(maxsub,maxsub),
     &                 tpk(maxsub,maxsub),ptpk(maxsub,maxsub),
     &                 thetr(maxsub,maxsub),thett(maxsub,maxsub),
     &                 dfvec(maxsub*maxvar+2*maxsub**2),
     &                 dfjx(maxsub*maxvar+2*maxsub**2),
     &                 movelo(maxsub,maxsub),
     &                 oldrho,dscp,dscn,lomax
```

```
      integer iref(maxsub,maxsub),is(maxsub),mmark(maxsub,maxcon),
     &        icheck,ictr,ictr2,itcnt,idcnt,i,j,ii,jj
      do 1 ii = 1,k
         alph(ii,ii) = 1.0
         beta(ii,ii) = 1.0
    1 continue
      do 2 ii = 1,k
         do 2 jj = 1,n(ii)
            prevx(ii,jj) = x(ii,jj)
    2 continue
      oldrho = rho
c     CALCULATION OF VALUES OF KS FN.
      call kpocal(k,rho,x,kpo,np,n,cmat,beta,d)
      do 3 i = 1,k
         do 3 j = 1,n(i)
            rhox(i,j) = 0.0
    3 continue
c     CALCULATION OF apk AND iref
c     iref IS AN ARRAY THAT IS USED TO INDICATE ZERO
c     CROSS DERIVATIVES AND IS USED TO SET THE LOWER AND UPPER
c     LIMITS ON THE r AND t COEFFICIENTS
      call apkcal(np,n,cmat,beta,x,rho,k,d,apk,iref)
c     INITIALIZATION OF r,t AND s COEFFICIENTS
      call rtsinit(kpo,np,n,cmat,beta,x,rho,rpk,k,d,apk,tpk,is,
     &           prpk,ptpk)
      icheck = 0
      dscp = 0.5
      dscn = 0.5
c     REPEAT THE PROCESS FOR THE SPECIFIED NUMBER OF ITERATIONS
      do 13 itcnt = 1,itval
c     SUBSPACE OPTIMIZATIONS
         do 4 i = 1,k
            call subopt(k,i,cmat,amat,d,x,rpk,is,tpk,kpo,newx,rho,n,np,
     &                  beta,alph,mlam,mmark,delx,prevx)
    4    continue
         do 5 ii = 1,k
            do 5 jj = 1,n(ii)
               prevx(ii,jj) = newx(ii,jj)
    5    continue
         ictr = 0
         ictr2 = 0
         call kpocal(k,rho,newx,kpo,np,n,cmat,beta,d)
         call supd(kpo,is,k,itcnt)
c     CALCULATION OF DERIVATIVES OF theta W.R.T r AND t COEFFICIENTS
         do 6 i = 1,k
            call calthet(i,cmat,d,newx,n,np,k,beta,thetr,thett,is,mlam,
     &                   mmark,ictr,ictr2,kpo)
    6    continue
         dscp = dscn
         idcnt = 1
         do 7 ii = 1,k
            do 7 jj = 1,n(ii)
               dfvec(idcnt) = newx(ii,jj)-x(ii,jj)
               dfjx(idcnt) = x(ii,jj)
               idcnt = idcnt+1
    7    continue
```

```
      do 8 i = 1,k
         do 8 jj = 1,n(i)
            x(i,jj) = newx(i,jj)
            if (dabs(x(i,jj)).le.1.0E-15) x(i,jj) = 0.0
 8    continue
c     TO CALCULATE THE OBJECTIVE FUNCTION VALUE
      call objcal(amat,x,k,n,sum)
c     ictr AND ictr2 INDICATE WHETHER THE COP NEEDS TO BE
c     SKIPPED i.e., IN THE CASE OF A CONSTANT OBJECTIVE FUNCTION
      if ((ictr.ne.k**2).or.(ictr2.ne.k**2)) then
         call apkcal(np,n,cmat,beta,newx,rho,k,d,apk,iref)
         do 10 ii = 1,k
            do 9 jj = 1,k
               if (apk(ii,jj).gt.tlim) then
                  lomax = -tlim
               else
                  lomax = -apk(ii,jj)
               endif
               movelo(ii,jj) = lomax
 9          continue
10       continue
c     COORDINATION OPTIMIZATION PHASE
         call copphase(k,thetr,thett,rpk,tpk,tlim,movelo,iref,prpk)
         if (itcnt.ge.itfix) then
            tlim = 0.8 * tlim
         else
            tlim = 1.0 * tlim
         endif
         do 11 ii = 1,k
            do 11 jj = 1,n(ii)
               dfvec(idcnt) = rpk(ii,jj)-prpk(ii,jj)
               dfjx(idcnt) = prpk(ii,jj)
               idcnt = idcnt+1
               dfvec(idcnt) = tpk(ii,jj)-ptpk(ii,jj)
               dfjx(idcnt) = ptpk(ii,jj)
               idcnt = idcnt+1
11       continue
         do 12 i = 1,k
            do 12 j = 1,k
               prpk(i,j) = rpk(i,j)
               ptpk(i,j) = tpk(i,j)
12       continue
      endif
c     TESTING THE CONVERGENCE CRITERION
      call chconv(dscp,dscn,CONTOL,idcnt,icheck,rho,oldrho,x,rhox,
     &            dfvec,dfjx,n,k,itcnt)
      if ((icheck.eq.1).or.(rho.gt.1.0D+7)) return
13    continue
      return
      end
```

VITA.

Jayashree Shankar was born on 2nd October, 1966 in Madras, India. She graduated from Good Shepherd Higher Secondary School in Madras, in April 1983. She received her first Bachelors Degree in Mathematics from Stella Maris College, Madras, affiliated to the University of Madras, in April 1986. Her second Bachelors Degree was in Computer Science from the Indian Institute of Science, Bangalore, in June 1989. In August 1992 she earned an MS degree in Computer Science from Virginia Polytechnic Institute and State University, Blacksburg, Virginia.