IMPROVED IHFEASIBLE PATH OPTIMIZATION FOR SEQUENTIAL MODULAR SIMULATORS PART II: THE OPTIMIZATION ALGORITHM

by

L.T. Rieglcr & J.E. Cuthrell

December, 19fH

DRC-O6-M2-R3

Improved Infeasible Path Optimization for Sequential Modular Simulators Part 2: The Optimization Algorithm

by

L.T. Biegler and J.E. Cuthrell

Working Paper - Preliminary Results

Recently, it was shown that chemical processes modeled by steady-state simulators could be optimized without repeatedly converging the process simulation. Instead, optimization and simulation of the flowsheet can be performed simultaneously, along an infeasible path, thus leading to much more efficient performance.

In this two-part study, we describe several improvements to this infeasible path approach. This second paper deals with theoretical and computational improvements to the optimization algorithm. Since recycle convergence and optimization occur simultaneously, the consequences of algorithmic failure can be severe, since little useful information is recoverable. Here we consider and improve several factors that affect the efficiency and robustness of the Successive Quadratic Programming (SQP) optimization algorithm.

The improvements are demonstrated on several small problems as well as three chemical process problems. The results show significant improvement in performance as predicted from the theory.

SCOPE

In Part one of this study we described the importance of process simulation for design and analysis. However, the present mode of simulation, the sequential modular strategy, can be very inefficient for optimization. Previous studies have shown (e.g. Gaines and Gaddy (1976), Ballman and Gaddy (1977)) that performing an optimization study by repeatedly simulating the flowsheet requires prohibitive computational effort, ~~it~~ in terms of demands made on the process simulator.

To make process optimization on sequential modular simulators more efficient and effective, Biegler and Hughes (1982) proposed the infeasible path approach. Here the process flowsheet and the optimization problem are converged simultaneously; the inefficient recycle convergence calculations are totally eliminated and the flowsheet itself is converged automatically at the optimum point. Several studies on a variety of test problems (Biegler and Hughes (1982,1983), Chen and Stadtherr (1983), Hutchison et al (1983)) have shown that this approach is much more efficient than previous optimization strategies. In this two part series we discuss further improvements to the infeasible path approach. Part one of this study (Biegler and Shivaram, 1983) deals with improving the interface between the process simulator and the optimization algorithm. The principal consideration there is the calculation of the objective and constraint functions and their gradients from the modules in the process flowsheet. Additionally optimal tearing strategies and non-differentiabilities in flowsheet modules were discussed. In this paper we deal exclusively with the optimization algorithm. Because the process flowsheet and the optimization problem are converged simultaneously by this algorithm, little information can be recovered if the algorithm fails. For this reason, the optimization algorithm must be efficient and robust. Specifically, it must be able to handle poor starting points, badly scaled problems and some inaccuracies in the gradients. In Part one of this study we mention that the SQP algorithm (Han (1977), Powell (1977)) has been an efficient and effective tool for optimization. In this paper we improve upon the efficiency and reliability of this algorithm.

Introduction

As mentioned in the Part one of this series, the infeasible path op* timization algorithm can be given by:

(NLP) $\min_{x} \phi(x)$

s.t. $g(x) \wedge 0$

$h(x) = 0$

where x - decision variables chosen by designer and tear variables

p - objective function

g - inequality constraints

h - equality constraints imposed by the designer and tear equations.

Part one discussed the formulation of this problem, the role of the variables in the process flowsheet and how constraint functions were evaluated.

The SQP algorithm solves the above problem by determining a search direction at each iteration i from the quadratic programming problem (QP):

Q(xi,B): $\min_{d} \wedge(x^1)^1 d + \frac{1}{j} d^T Bd$

s.t. $gU^1) + v\, gU^1)^i d \pounds o$

$Mx^1) + v\, h(x^i)^T d = 0$

Gradients for Q(x*,B) are generally obtained from the flowsheet by perturbation. The Hessian matrix, B_r is constructed through BFGS (see Dennis and More (1977)) updates which rely on gradients of the Lagrangian: $L(x,u,v) = \#(x) + u^Tg(x) + v^Th(x)$ at successive points (here u and v are Kuhn-Tucker multipliers returned from $Q(x^i{}_rB)$).

Once Q(x*,B) determines a search direction, d_r a stepsize, \ , along this direction must be chosen to set the next point, i.e. $x^{i+1} =$ x* •+ \d. In order to choose this step a sufficient decrease in some merit function must be found along the search direction. This function must include an objective function term plus a nonlinear combination of the violated constraints^ To achieve desirable performance the stepsize or line search procedure must avoid determining steps that are either so large that they lead to cycling or divergence, or so small that the rate of convergence becomes undesirably slow.

Finally, while the theoretical convergence properties of the SQP algorithm are determined by implementation of the quadratic program and stepsize procedure, the most drastic effect on performance is produced by the scaling of the optimization problem. While several automatic scaling algorithms have been proposed, (see Tomlin (1975)) none has been universally effective in improving performance.

This paper deals with three areas of improvement relating to the SQP algorithm. Taken together, they represent an improvement in the computational effort required by the propylene chlorination optimization (see Part one) by up to a factor of *-jr^re^* The three areas are:

1) Improvements in setting up and solving Q(x*,B).
2) More efficient and robust line search procedures.
3) An automatic scaling procedure and an 'on-line measure of how well the problem is scaled.

Each area is motivated by examples that illustrate serious problems with existing strategies. We also present, in the final two sections extensive numerical results that illustrate the effectiveness of our improvements.

I. The Quadratic Programming Step.

The most basic step in the SQP algorithm is formulation and solution of the quadratic program, Q(x*,B). By analogy, one can compare this step to Gaussian elimination of the Jacobian in a Newton Raphson procedure. Both cases strongly depend on the accuracy and efficiency of the basic steps. Aside from the effort needed to evaluate function values and gradients, this step is the most time-consuming in the SQP algorithm.

Perhaps the most widely used version of the SQP algorithm is VF02AD, a FORTRAN-callable subroutine in the Harwell Library. Developed by Powell (1977), it uses a QP algorithm, VE02AD developed by Fletcher (1971)• At present, however, VE02AD is generally regarded as inefficient because the constraint updating procedure for the active set is time consuming and often has a tendency to become unstable. Several implementations (Locke et.al (1983), Stadtherr and Chen (1983)) as well as our own use a QP algorithm developed by Gill and Murray (1978). This algorithm performs a QR factorization of the active constraint normals and optimizes only in the linear subspace of the active set. In addition, Gill and Murray developed constraint updating formulae that apply directly to the factorizations, are exact, and do not contribute to the growth of roundoff error. Thus their algorithm incorporates a stable and efficient updating procedure for solving QP'^f^s.

Another provision allowed by the Gill and Murray QP algorithm is the use of "warm" starts. Much of the solution time for solving QP's is spent in determining the active constraint set that satisfies Kuhn-Tucker conditions. The "warm" start provides a "good guess" of the active set before the solution procedure begins. Since the SQP algorithm often chooses its active constraints in the first few iterations, the active set for the QP seldom changes from iteration to iteration. Thus, using "warm" starts at each iteration greatly reduces the QP solution time. Both Locke et al. (1983) and Stadtherr and Chen (1983) eliminate the equality constraints and reduce the QP before solving it. With Gill and Murray's QR factorization of the constraint normals and the use of "warm" starts, it is easy to see that removing equality constraints does not necessarily lead to a more efficient algorithm on small problems.

Even with an efficient QP procedure, the SQP algorithm can sometimes be plagued with inconsistent constraint linearizations. Here the local linearization of the nonlinear constraints produces an empty feasible region. Since no feasible point exists, the QP algorithm fails. To combat this problem, Powell (1977) added an extra parameter |- to Q(x*-,B) to form:

$$Ql\ (xi,B):\quad \underset{d}{\text{Min}}\quad J(xi)^T d\ +\overset{1}{J}\ d^T B d\ +\ T)|$$

$$\text{s.t.}\quad \text{maxtOrg}\wedge x^1)]\wedge +\ \min[O,gj\ (x^1)]\ +\ v\ 9j\ (x^1)^*\ d\ \ £0$$

$$h(x^l)\ I\ +\ 7\ \ Mx^1)^1\ d\ =\ 0$$

$$0\ *\ §\ *\ 1\ ,\quad Tl\ =\ -10^6$$

For QP^{f}s that have no feasible region, the heuristic incorporation of $§_r$ tends to shift the constraints and "create" a feasible region. (By specifying d = 0 and 5 = 0, one automatically specifies a feasible point for Q£jx*-,B). In his code, VF02AD, Powell terminates the SQP algorithm if § = 0 is ever a solution to Q g (xi,B). Here he assumes no feasible point can be found for the (NLP).

While this procedure is conceptually attractive it suffers from two serious deficiencies. First, consider the problem:

$$\underset{x}{\text{Min}}\quad x_2$$

$$\text{s.t.}\quad 1.0 + x_1 - x_2^2\ £\ 0$$

$$1.0 - x_1 - x_2^2\ £\ 0$$

$$x_2 * 0$$

shown in Figure 1. From the figure it is clear that the solution lies at point B. Starting at point A, however, it is easy to see that the constraint linearization is inconsistent and the solution to Q§ (x^,B) contains § = 0. Powell's procedure thus terminates at point A because it assumes no feasible point can be found. Second, while Q§ (x*,B) often finds a solution when Qtx^B) has no feasible region, it does not **follow** that Q ^(x^B) and Q(x^A,B) give the same solutions if a feasible region exists. For example, the simple quadratic program:

$$\min_{d} \quad d_1 + d_2 + 5.0d_1^2 + 5.0d_2^2$$

$$\text{s.t.} \quad d_1 + d_2 = 1001$$

$$d_1 - d_2 = 999$$

has the solution $d^T = [\ 1000, 1\ ]$. If we augment this QP to form $Q\xi\ (x^i, B)$ then the solution is $d^T = [\ 100.1,\ 1.001\ ]$ and $\xi = 0.1001$. Thus the procedure proposed by Powell may interfere with the true solution of the QP.

In our implementation we simply solve $Q(x^i,B)$ at every iteration. If we encounter an inconsistent constraint linearization the QP routine of Gill and Murray terminates but also calculates the minimum infeasibility (MI) for $Q(x^i,B)$. At this point we reset the maximum constraint violation tolerance to (1.01) MI and solve the QP again. In this way a feasible region is "created" and a search direction is found that minimizes the transformed QP. This procedure is conceptually similar to one recently developed by Tone (1983) which includes slack variables in the constraints.

While this simple procedure is not guaranteed to handle all inconsistent linearizations, it clearly solves the two examples given above. In the first example, a feasible region is created that allows movement from point B to point A. The second example presents no problem since $Q(x^i,B)$ has a solution. After considerable experience on a number of test problems, we found this procedure superior to the one proposed by Powell.

II. The Line Search Procedure

Han (1976) and Garcia-Palomares and Mangasarian (1976) showed that the SQP algorithm has a superlinear convergence rate if:

a) the starting point is sufficiently close to the solution,

b) full steps are taken along the search direction, and

c) the Hessian is calculated using rank-two quasi-Newton updates such as DFP or BFGS (see Dennis and More (1977)).

Han (1977) showed that if the stepsize, λ , is chosen by reducing an exact penalty function:

$$P(x,\alpha) = \phi(x) + \alpha \left[\sum_{j=1}^{m} g_j(x)_+ + \sum_{j=1}^{meq} |h_j(x)| \right]$$

where $g_j(x)_+ = \max[\ 0\ ,\ g_j(x)\ ]$

$$\alpha = \| u , v \|_\infty$$

•along the search direction d, then the SQP algorithm converges to a Kuhn-Tucker point from any starting point (global convergence). However, using this line search procedure often leads to very small stepsizes and slow convergence rates, especially in the neighborhood of the solution (Maratos, 1978). To try to relax the stepsize procedure, Powell (1977) introduced a less stringent line search function:

$$P_p(x,\alpha) = \phi(x) + \sum_{j=1}^{m} u_j\, g_j(x)_+ + \sum_{j=1}^{meq} v_j\, |h_j(x)|$$

where at iteration i:

$$u_j^i = \min \left[u_j^{i-1} , \tfrac{1}{2} (\bar{u}_j + u_j^{i-1}) \right]$$

$$v_j^i = \min \left[v_j^{i-1} , \tfrac{1}{2} (\bar{v}_j + v_j^{i-1}) \right]$$

This function, however, possesses neither global nor local superlinear convergence properties. Chamberlain et al (1982) showed that this function can exhibit the same slow convergence behavior that the exact penalty function does. Moreover, Chamberlain (1979) showed that the SQP algorithm, with Powell's line search function, cycles between points A and B for the problem given in Figure 2. The optimum for this problem is clearly at point C.

To remedy these problems, Chamberlain et al (1979) proposed the "watchdog" technique. Here the stepsize is chosen by reducing either the Lagrangian: $\phi(x) + u^T g(x) + v^T h(x)$ or the exact penalty function during the line search. The convergence properties, however, **require** a reduction in the exact penalty function every t iterations (where $t \geq 2$). If no reduction occurs, the algorithm must restart from a previous point. A simplification of this algorithm was recently implemented by Stadtherr and Chen (1983) and found to be computationally more efficient than VF02AD. The studies of Biegler and Hughes also used the "watchdog" algorithm with the exception that a modified Lagrangian: $\phi(x) + u^T g(x)_+ + v^T h(x)$ was used so that feasibility of the inequality constraints is not rewarded. More recently, Chamberlain et al (1982) modified the original watchdog algorithm; they simply [illegible] alternate between taking full steps along the search direction ~~and~~ or reducing the exact penalty function. Again, the convergence properties of this algorithm are valid since the exact penalty function must still be reduced every t iterations and a restart is required if no reduction occurs.

Several investigators considered other line search strategies for the SQP algorithm. Fletcher (1982) and Mayne and Polak (1981) sought to remedy the "Maratos effect" by proposing second order correction search arcs to the search direction in the neighborhood of the solution. However, it is difficult to determine when to apply these corrections before the solution is known. Schittkowski (1981a) and Yamashita (1982) proposed an augmented Lagrangian:

$$L_a(x,u,v,\alpha) = \phi(x) + \frac{1}{2\alpha} \sum_{j=1}^{m} \left[(\alpha\, g_j(x) + u_j)_+^2 - u_j^2 \right]$$

$$+ v^T h(x) + \frac{\alpha}{2} h(x)^T h(x)$$

for the line search. Both authors showed that an SQP algorithm using an augmented Lagrangian line search function has local superlinear and global convergence properties under certain conditions. However, numerical experience reported by Schittkowski (1981b) showed this line search function to have inferior performance to Powell's line search function. To explain this, he mentions that his procedure for updating the penalty parameter, *a* , often causes it to tend to infinity.

In this section we introduce a new line search function:

$$L^*(x,u,v,<y) = 0(x) + u^T g(x)_+ + v^T h(x) + | Hg(x)_+ , h(x)|l^2$$

Cuthrell and Biegler (1983) showed that for an equality constrained problem, the solution of Qtx^B) is equivalent to taking a truncated Newton step for V L (x,v,or) = 0 in the space of the variables (x_fv). Thus L (x,u,v,or) is a natural line search function for an SQP algorithm. It was shown that for certain values of a # the SQP algorithm with L^*_(x,u,v, *a*) as the line search function is globally and locally superlinearly convergent. Moreover, the function can be factored for or , which allows us to determine a region for or where the line search test is satisfied and conditions for convergence are met. If this region does not exist, a smaller step must be taken.

Cuthrell and Biegler (1983) have also shown that if the solution of the QP gives a descent direction for L^*(x,u,v,or)r then the SQP.algorithm is globally convergent. Writing this condition, V L (x,u,v,a) ⁻ p < 0,

where $p=\begin{bmatrix} \overset{d}{} \\ u - u^i \\ \bar{v} - v^1 \end{bmatrix}$ is the solution of Q(x ,B), in terms of X yields:

$$= \frac{\nabla\phi(x^l)^T d + (\bar{u} - 2U^1)\ gCx^1)_+ + (\bar{v} - 2V^1)\ h(x^i)}{11 8(x)_{+,} >^h (^x) 11}$$

The property for a sufficient decrease during a line search given by Armijo(1966) is:

$$L^*(x^1 + Xd , u^1 + X(\bar{u} - u^1) , v^1 + X(\bar{v} - v^1) , a) \wedge$$

$$L^*(x^1, u^1 , v^1 , a) + X « v L^*(x^1 , u^1 , v^1 , cr)^T P$$

As seen in Figure 3, the line search is satisfied when the value of L* for a given X lies below the chord specified by $L^*(z^1)$ and $\nabla L^*(z^1)^T p$ Since the terms in the inequality can be factored for a, we can derive the bound:

$$"LS \quad \frac{L^\wedge(z^1 + X) - L.(z^1) - X6\ 7L,(z^1)^\wedge}{T(X)}$$

where

$$L_+(z^i) = \phi(x^i) + u^T g(x^i)_+ + v^T h(x^i)$$

$$\nabla L_+(s^-)^A{}_p = V\ J((x^-)'_p + (u - 2u^-)^1 g(x^-)_+ + (v - 2v^-)^\wedge$$

$$Y(\lambda) = - \backslash \ II\ gCx^1 + Xd)_+ ,\ hCx^1 + Xd)|l^2 + (y_2 - X6)U\ g(x^i) + ,\ h(x^i)||^2$$

Here Of_{LS} is either a lower bound (if T (X)>o) or a*upper bound (if T (X)<0) on a which if satisfied for the current \ , the Armijo condition will be satisfied. The bounds or_{LS} and ct_{dd} can be used to adaptively choose the penalty parameter cr and results in fehe following line search procedure:

1) Set X = 1

2) a^1 = max (0 , $cr^*_t|_d$ + l(f^3

3) If Y(X) > 0 (U a£g > a^i_d go to 5)

Else, continue.

4) If LV + X_p) ^ L* (z^1) + X6 7 L*(z)^{T}P 8o t0 5)

Else$_f$ determine a smaller \ by say quadratic interpolation and go to 3)

5) Update according to $z^{i+1} \leftarrow z^i + X P$

i - i + 1

The line search procedure terminates and Q(x* ,B) is solved at the new point.

In step 2 we require α to always be positive, since infeasibility would be rewarded for $\alpha < 0$, and that $\alpha > \alpha_{dd}$ to ensure a descent direction. To understand step 3 first recall that for $\Psi(\lambda) > 0$, α_{LS} is a lower bound on α. Thus either $\alpha_{LS} < \alpha_{dd} < \alpha$ or $\alpha_{dd} < \alpha_{LS}$. In the former case we satisfy both the descent property and Armijo test, while for the latter case α must be increased before the Armijo test can be met. Rather than resetting α explicitly, we can just exit the line search procedure for the current stepsize and thereby make no unnecesary restrictions on subsequent iterations. Thus for $\Psi(\lambda) > 0$ the line search can be satisfied. Recall also that α_{LS} is an upper bound for $\Psi(\lambda) < 0$ leaving again two cases, $\alpha_{dd} < \alpha_{LS}$ or $\alpha_{LS} < \alpha_{dd} < \alpha$. For the former case both requirements are again satisfied. Only for the latter case must the stepsize be reduced since α cannot simultaneously satisfy each condition. Thus for $\alpha_{LS} > \alpha_{dd}$ we may also accept the current stepsize, λ.

One further comment on step 2 must be made. To rigorously ensure a descent direction at each iteration, the condition $\alpha^i > \alpha_{dd}^{max}$ where $\alpha_{dd}^{max} = \max\{\alpha_{dd}^i\}$ i=1,2,3... must be enforced. This requirement is however overly restrictive since α_{dd} often reaches its maximum value during the first few iterations and was significantly less for the remainder of the solution processs. This led to unnecessary restrictions during later iterations and sometimes resulted in the taking of small steps. Step 2 in the above algorithm represents a mild relaxation since it only requires $\alpha^i > \alpha_{dd}^i$ for each i. We could essentially obtain the same result by simply restarting the problem from a point subsequent to where the large α_{dd} occurred.

The above algorithm was tested on fifteen nonlinear programming problems listed in Table 1. Here the letter and number corresponding to each problem indicates the reference for the problem and the problem number in the reference. The number of variables (N), total number of constraints (M) and number of equality constraints(MEQ) are also listed for each problem. The following algorithms were compared:

OPT - SQP with the above augmented Lagrangian line search procedure given and the Gill and Murray (1978) QP algorithm

OPTHP - same as OPT except with Powell's line search procedure

WDOG - SQP algorithm used in Part one of this study (Biegler and Shivaram (1983)) with Fletcher's (1971) QP program and the watchdog line search procedure.

The results show that OPT never requires more function evaluations than OPTHP. On the first two problems, given by Chamberlain (1979), both OPTHP and WDOG fail to converge; they oscillate continually between two infeasible points. For example, for Figure 2 which depicts problem Al, OPT starts at point A and converges to the optimum, point C, in 3 iterations; the other algorithms simply cycle between A and B. OPT is also generally faster than WDOG although nothing in the theoretical development guarantees this. Note that on most problems, all three algorithms required about the same number of function evaluations. For these cases, OPT and OPTHP obviously have equivalent CPU times because the only difference was in the line search procedure. WDOG, on the other hand, which used a less efficient QP solver, failed on three out of 15 problems, and required over 50% more CPU time than OPT on the problems it solved successfully.

OPT was then compared with WDOG on the three chemical process problems described in Part one of this study. Here the scale factors, perturbation sizes and convergence tolerances were the same as in Part one. The results are given in Table 2 and illustrate the difference in performance between the two algorithms. For the first problem since both algorithms take full steps to the solution, no conclusions can be reached. On the second problem the new line search procedure actually required more functions evaluations to solve the problem. This is easily explained be analyzing the iterates in terms of the objective function countours, (see Figure 4 in Biegler and Hughes (1982)). The restrictive WDOG algorithm luckily finds a point near the ridge, by taking less that a full step, and then proceeds quickly to the solution• OPT, on the other hand, takes a full step which terminates well over the ridge and then must spend time moving back. Problem 3 clearly demonstrates the effectiveness of the new procedure by requiring slightly less than half the number of function evaluations and thus significantly less CPU time. Even though WDOG uses a less efficient QP solver the CPU time reduction time is still clear. WDOG also terminated due to line search failures while OPT terminated normally (this indicates that gradient error doesn't allow convergence to so tight a tolerance).

The new algorithm, again, is not guaranteed to perform better than WDOG as is seen by both starting points for problem 2. However, it seems to be more reliable and effective that WDOG on a large number of problems, and also exhibits some desirable convergence properties.

III. Scaling Algorithms for SQP

Perhaps the least understood and most important part of process optimization is the appropriate formulation and scaling of the original problem. At present, there are no foolproof scaling criteria; the best choice of the scale set is usually problem dependent and often determined by experimentation. To study this problem and provide some general guidelines for the infeasible path strategy we note that for the SQP algorithm:

1) The quadratic program is scale invariant under changes in constraint or variable scaling.

2) The BFGS update is also scale invariant under linear transformations of the functions or variables.

These two statements mean that neither updates of the Hessian approximation nor the QP solution are affected by the scale set if exact arithmetic is used. However, the following reasons indicate why scaling can greatly influence the SQP algorithm:

1) Since the SQP is nothing more than a Quasi-Newton method applied to the gradient of L with respect to both x, and

the multipliers of the active constraints, the initial Hessin B should be:

$$7_{xx} L(x°,u°,v°) = V^2 flj(x°) + u°V^2 g(x°) + v° 7^2 W°)$$

Since second derivatives are not available in SQP, no information is available for the initial approximation of B.

2) Of course, inaccurate gradients, calculations not done in exact arithmetic and the use of bad pivot sequences lead to the accumlation of rounding errors which often result in inaccurate QP solutions.

The first statement is probably the main reason why the SQP algorithm is sensitive to variable scaling. In most implementations B is initially set to the identity matrix because no further information is available. Scaling the variables simply changes the initial Hessian approximation to another diagonal matrix.

Here: $\bar{x} = Cx$ $\bar{B} = C^{-1} BC^{-1}$

where c is a diagonal matrix

How well this matrix approximates 7_{xx} L(z°) and the nonlinear surface determines the performance of the algorithm. However, this cannot be determined a priori.

Several scaling algorithms have been proposed for the SQP algorithm. The documentation for VF02AD recommend scaling the gradients to "around one" to avoid line search failures. Biegler and Hughes (1982) recommended as a first guess, to scale the variables so that the gradients of the objective function have elements with absolute values between ten and one-hundred. Stadtherr and Chen (1983) simply scaled the Hessian matrix based on values of rf(x°), x° and 7$(x°). However, none of these "automatic" methods consistently give better performance even when compared to unsealed problems.

In this section we develop a scaling algorithm based on the upper and lower bounds of the design and tear variables. As with any automatic scaling method, we cannot guarantee improved performance for all problems. Instead this method provides a set of guidelines for scaling process optimization problems solved by the infeasible path strategy.

To scale the variables we note that the solution to $Q(x^x,B)$ is in large part determined by the equality constraints, $h(x) = 0$, that are given by the tear equations. Since the gradients of these equations are directly related to the magnitude of the variables, we simply scale the variables x so that they are bounded by 0 and 1. To determine the scale factor we use the suggestion by Tomlin (1975) that all scale factors be integer powers of the floating point base (in this case, 2). Choosing the bounds of x as our variable scale gives:

$$\bar{x} = Cx$$

$$C_{jj} = 2^{-a} \qquad a = \mathrm{int}[\ \log_2 (x_u - x_\ell)\ ]$$

It is important to mention that the variable bounds must be physically meaningful. Normally, the designer has a good idea over what range of design variables the model should be optimized. To a lesser extent he has some idea of the range of tear variables. Thus, the variable scale factors should reflect fairly accurately the order of magnitude of x. Obviously, specifying bounds of plus and minus infinity makes no sense.

In addition, the constraints in the QP must be scaled in order to prevent inaccuracy in the pivoting step. In this case we simply use the initial values of the constraints as scale factors, provided they are not close to zero. If the constraint is below a zero tolerance (e.g. 10^{-3}) the scale factor is set to one. Otherwise:

$$\begin{bmatrix} \bar{g} \\ \bar{h} \end{bmatrix} = R \begin{bmatrix} g \\ h \end{bmatrix}$$

$$R_{jj} = 2^{-a}$$

$$a = -\mathrm{int}\,[\log_2(|g_j^o| \text{ or } |h_j^o|)]$$

This simple procedure provides only a suitable normalization procedure. After this step the engineer may (and probably should) perform some further scaling based on his experience and insight. In this paper, we demonstrate that even with this simple algorithm some improvement can be obtained for the infeasible path algorithm.

Consider the fifteen test problems solved in Table 1. Of these, seven have meaningful upper and lower variable bounds specified in their problem statements. These were solved using the above scaling algorithm and are listed in Table 3. Compared to the unscaled results, the scaled algorithm never required more function evaluations, although scaling leads to improvement on only two of them. The greatest improvement occurs with problem D4, the alkylation problem of Bracken and McCormick (1968). Note here that the equality constraints play the same role as the tear equations in infeasible path optimization.

In this comparison three scaling procedures were compared to the unscaled process problems. The first scale set (OPTSCALE) was determined by experience, after running the three process optimization problems several times. These scale sets were used in Part one of this study and were determined in previous studies (see Biegler and Hughes (1982,1983)). The second set of scale factors were derived from the heuristic proposed in Biegler and Hughes (1982), that $|\nabla \phi(x_j)|$ be scaled between ten and one hundred. Finally, the third set of scale factors (New Scale) was derived from the algorithm above. The first two methods led to scale factors in powers of ten while the third method yielded scale factors in powers of two.

The scaling results are given in Table 4. Note that the unscaled infeasible path algorithm took small steps on the first two problems and terminated before reaching the optimum. Imposing a Kuhn-Tucker tolerance tighter than 10^{-3} may improve these solutions although with the given perturbation sizes for gradient evaluation, line search failures may be encountered first. For the unscaled algorithm the third problem terminates reasonably close to the optimum after 39 iterations.

Lastly, we consider the conditioning of the QP problem. As pointed out above, one of the reasons for constraint and variable scaling is to avoid the buildup of roundoff error through inaccurate gradients and bad pivot sequences. In order to judge the effectiveness of scaling, it is useful to monitor the conditioning of the problem and if necessary, rescale the problem if it becomes ill-conditioned.

To develop a measure for conditioning, consider the QP step. The QP solution d is given by the linear equations $Bd = -\nabla L(x^*, \bar{u}, \bar{v})$. Here the relative error in the solution d is given by:

$$\frac{\|\delta d\|}{\|d\|} \le \kappa(B) \quad \frac{\|\delta \nabla L\|}{\|\nabla L\|} + \frac{\|\delta B\|}{\|B\|}$$

where $\|\cdot\|$ - any matrix or vector norm

δd, δB - absolute errors in d and B, respectively

$\kappa(B) = \|B\| \, \|B^{-1}\|$, the condition number of B

The condition number, $\kappa(B)$ thus indicates how much the error in the gradients is magnified in the solution, d. To keep the QP well-conditioned, $\kappa(B)$ must be kept low (say $<10^6$). To calculate $\kappa(B)$ we merely parallel the quasi-Newton BFGS update for B:

$$B^{i+1} = B^i - \frac{B^i s \, s^T B^i}{s^T B^i s} + \frac{y \, y^T}{y^T s}$$

with the inverse BFGS update:

$$(B^{i+1})^{-1} = \left[I - \frac{s\,y^T}{s^T\,y} \right] (B^i)^{-1} \left[I - \frac{y\,s^T}{s^T\,y} \right] + \frac{s\,s^T}{s^T\,y}$$

where $s = x^{i+1} - x^i$

$y = \nabla L\,(x^{i+1}, \bar{u}, \bar{v}) - \nabla L\,(x^i, \bar{u}, \bar{v})$

The condition number of the symmetric B matrix is then computed by taking the maximum row (or column) sums of $|B_{ij}|$ and $|B_{ij}^{-1}|$ as norms.

Table 4 shows the maximum condition munbers for the different scaling procedures. Note that while there is not a strong correlation between performance and condition number, line search failures were observed for very high condition numbers (say $>10^{30}$). Also note that the new scaling procedure keeps K(B) relatively low. Based on these limited results, it seems that the new scaling algorithm performs surprisingly well for an a priori scaling procedure and serves as a good initial scaling method.

If the condition number becomes too high over the course of the optimization, a rescaling procedure can be implemented to make the problem better conditioned. Applying several scaling methods to the infeasible path algorithm, Xu (1982) reported significant improvements in performance if the problem is rescaled once the condition number becomes too high. To do the rescaling, several heuristic and rigorous methods are available (see Tomlin (1975), Bauer (1963)) for reducing the condition number. However, if B becomes too ill-conditioned, it may be advantageous simply to restart the Hessian approximation with the identity matrix.

CONCLUSIONS AND SIGNIFICANCE

This paper forms the second part of a study detailing improvements for the infeasible path optimization algorithm using sequential modular simulators. Here we concentrate on improvements to the successive quadratic programming (SQP) algorithm.

The improvements are divided into three areas. The first section deals with solution of the quadratic programming problem (QP) that determines the search direction at each iteration. We briefly discuss a new (QP) algorithm by Gill and Murray (1978) and its advantages over the commonly used Fletcher (1971) algorithm. We also develop a procedure for recovering from inconsistent constraint linearizations, for which the QP has. no solution, and demonstrate its effectiveness over an existing procedure suggested by Powell (1977).

The second section deals with the line search algorithm which determines a stepsize along the search direction. Current procedures based on exact penalty functions (see Powell, 1977) can cycle or converge very slowly on certain problems. Thus, present implementations of the SQP method usually contain line search procedures that possess neither local superlinear nor global convergence properties and therefore may exhibit undesirable performance. Here we present a method based on a modified augmented Lagrangian that has the above convergence properties. It performs significantly better than current strategies on fifteen well-known nonlinear test problems. We also apply this procedure on the process optimization problems described in the first part of this study (Biegler and Shlvaram (1983)) and demonstrate significant improvement on these as well.

Lastly, we consider scaling procedures for the SQP algorithm. In this section we develop a very simple procedure based on the variable bounds in the optimization problem. . While we make no claims as to its efficiency over scale sets determined by insight and experience, we see that the procedure serves as a very good initial scaling method. On the nonlinear test problems we observe significant improvement over unsealed runs. On the process optimization problems the new scaling procedure performs competitively with a scale set determined by experience. To monitor, over the course of the optimization run, the conditioning of the QP, which directly influences the search direction calculation we develop a method to efficiently calculate the condition number of the Hessian. This allows us to periodically rescale the problem if it becomes ill-conditioned.

The above improvements are based oh theoretical and computational insights. They result in better performance up to a factor of •f"/!*-jree compared to the results reported in Part one of this study. Because these improvements deal solely with the SQP algorithm, they are not restricted to infeasible path process optimization but have wide applicability for solving general nonlinear programming problems.

References

Armijo, L., "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives," Pacific J. Math., 16, p.l (1966)

Ballman, S.H. and J. L. Gaddy, "Optimization of Methanol Process by Flow-sheet Simulation," IEC Proc. Des. Dev., 16, 3, p.337 (1977)

Bauer, F., "Optimally Scaled Matrices," Num. Math. 5, p.73 (1963)

Biegler, L. T. and R. R. Hughes, "Infeasible Path Optimization of Sequential Modular Simulators," AIChE J., 28, 6, p.994 (1982)

Biegler, L. T. and R. R. Hughes, "Process Optimization: A Case Study Comparison," Comp. and Chem. Engr., to appear (1983)

Biegler, L. T. and S. Shivaram, "Improved Infeasible Path Optimization for Sequential Modular Simulators," Part 1: The Interface," submitted to AIChE J., (1983)

Bracken, J. and G. P. McCormick, _Selected Application of Nonlinear Programming_, Wiley, New York (1968)

Chamberlain, R. M., "Some Examples of Cycling in Variable Metric Methods for Constrained Minimization," Math Prog., 16, p.378 (1979)

Chamberlain, R. M., C. Lemarechal, H. C. Pedersen, and M. J. D. Powell, "The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimization," DAMTP 80/NA1, Unversity of Cambridge, (1979)

Chamberlain, R. M., M. J. D. Powell, LeMarechal, and Pedersen, "The Watchdog Method for Forcing Convergence in Algorithms for Constrained Optimization," Math. Prog. Study 16, p.l (1982)

Chen, H-S and M. A. Stadtherr, "Strategies for Simultaneous Modular Flow-sheeting and Optimization," 2nd International Conference on Foundations of Computer Aided Process Design, Snowmass, CO,(1983)

Colville, A. R.,"A Comparative Study on Nonlinear Programming Code" IBM New York Scientific Center Report #320-2949, (1968)

Cuthrell, J. E. and L. T. Biegler, "Augmented Lagrangian Line Searches for Successive Quadratic Programming," submitted to J. Opt. Theory Appl.,(1983)

Dennis, J. E. and J. J. More, "Quasi-Newton Methods, Motivation and Theory," SIAM Review, 19, 1, pp.46-89 (1977)

Fletcher, R., "Second Order Corrections for Nondifferentiable Optimization," Lecture Notes in Mathematics 912, Springer Verlag, Berlin (1982)

Garcia-Palomares, u. M. and O. L. Mangasarian, "Superlinearly Convergent Quasi-Newton Algorithms for Nonlinearly Constrained Optimization Problems," Mathematical Programming, 11, p. 1-13 (1976)

Gill, P. E. and W. Murray, "Numerically Stable Methods for QP," Math. Prog., 14, p.349 (1978)

Han, S-P, "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," Math Prog., 11, p.263 (1976)

Han, S-P, "A Globally Convergent Method for Nonlinear Programming," J. Optimization Theory and Applics., 22, 3, p.297 (1977)

Harwell Subroutine Library, Atomic Energy Research Establishment, Harwell, UK, February, 1971

Himmelblau, D. M., Applied Nonlinear Programming, McGraw-Hill, New York,(1972)

Hutchison, H. P., S. Kaijoluoto, and W. Morton, "Process Optimization Usinga a Serial Cyclic Flowsheet Simulator," 3rd International Congress on "Computers and Chemical Engineering," Paris, (1983)

Locke, M. H., R. H. Edahl and A. W. Westerberg, "An Improved Successive Quadratic Programming Optimization Algorithm for Engineering Design Problems," submitted to AIChE J., (1983)

Maratos, N., "Exact Penalty Function Algorithms for Finite Dimensional and Conttrol Optimization Problems," Ph.D. Thesis, University of London (1978)

Mayne, D. Q. and E. Polak, "A Superlinearly Convergent Algorithm for Constrained Optimization Problems," presented at Dundee Numberical Anal. Conf. (1979)

Powell, M. J. D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," presented at the 1977 Dundee Conference on Numerical Analysis (1977)

Rosen, E. M. and S. Suzuki, "Construction of Nonlinear Programming Test Problems," Comm ACM, 8,p.113 (1965)

Schittkowski, K., The Nonlinear Programming Algorithm of Wilson, Han and Powell with an Augmented Lagrangian Type Line Search Function, Part 1: Convergence Analysis," Numer. Math., 38, p.83 (1982)

Schittkowski, K., "Part 2: An Efficient Implementation with Linear Least Squares Subproblems," Numer. Math., 38, p.115 (1982)

Schuldt, S. B., "A Method of Multipliers for Mathematical Programming with Equality and Inequality Constraints," JOTA, 17, 1, p.155, (1975)

Stadtherr, M. A. and H.S. Chen, "Numerical Techniques for Process Optimizationa by Successive Quadratic Programming," on "Computers and Chemical Engineering," Paris, (1983)

Tomlin, J. A., "On Scaling Linear Programming Problems," Math. Prog. Study, 4, p. 146, (1975)

Tone, K., "Revisions of Constraint Approximations in the Successive QP Method for Nonlinear Programming Problems," Math. Prog.,26, p.144 (1983)

Xu, C. Y., "Conditioning Test on SPOSEQ," Technical Report, Eng. Expt. Station, Univ. of Wisconsin (1982)

Yamashita, H., "A Globally Covergent Constrained Quasi-Newton Method with an Augmented Lagrangian Type Penalty Function," Math. Prog., 23, p. 75 (1982)

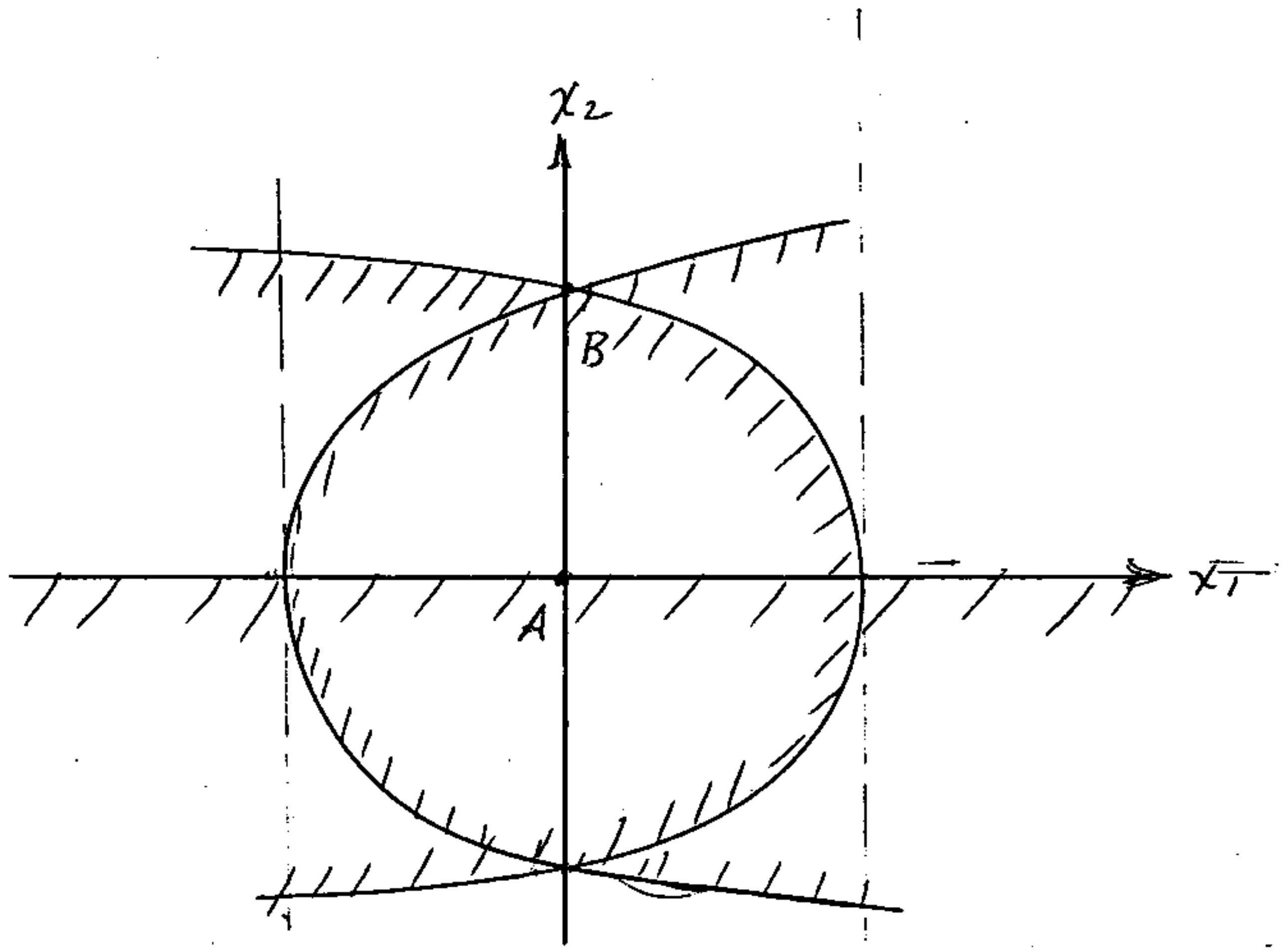
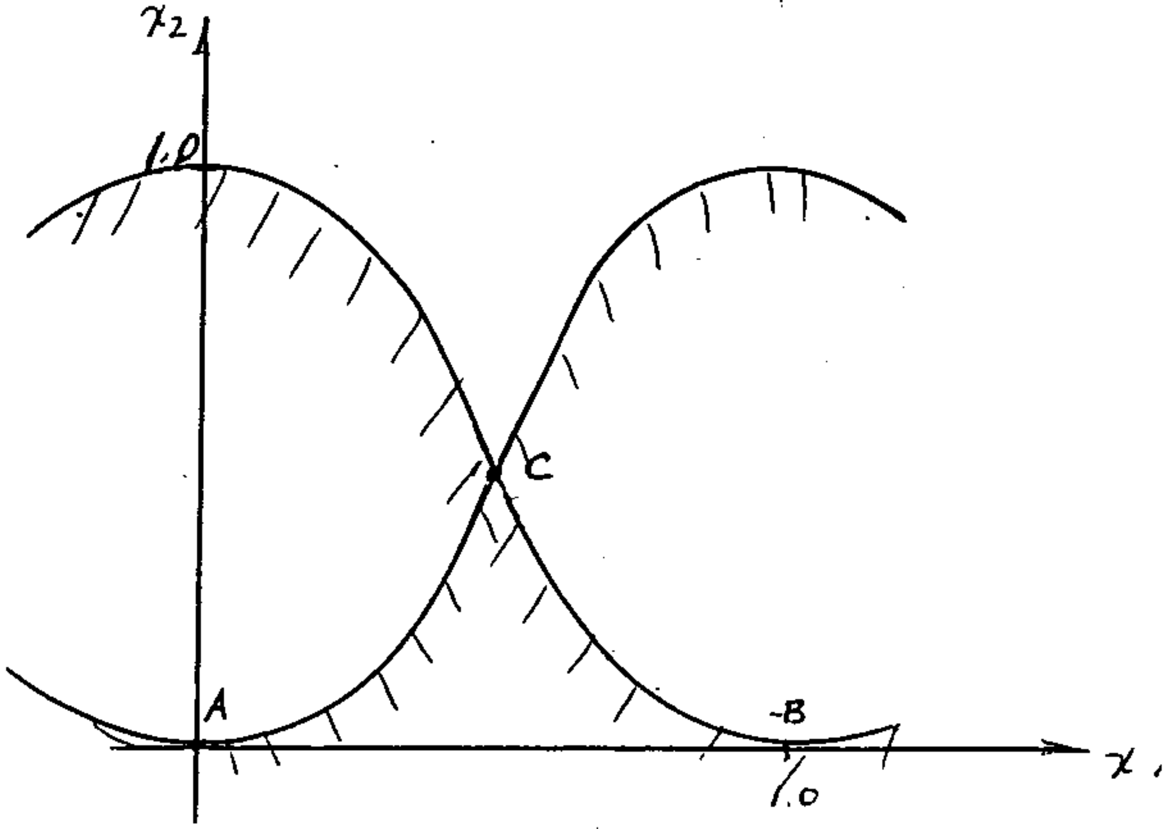


Figure 1

Inconsistent Constraint Linearizations for QP Problems



Min x_2

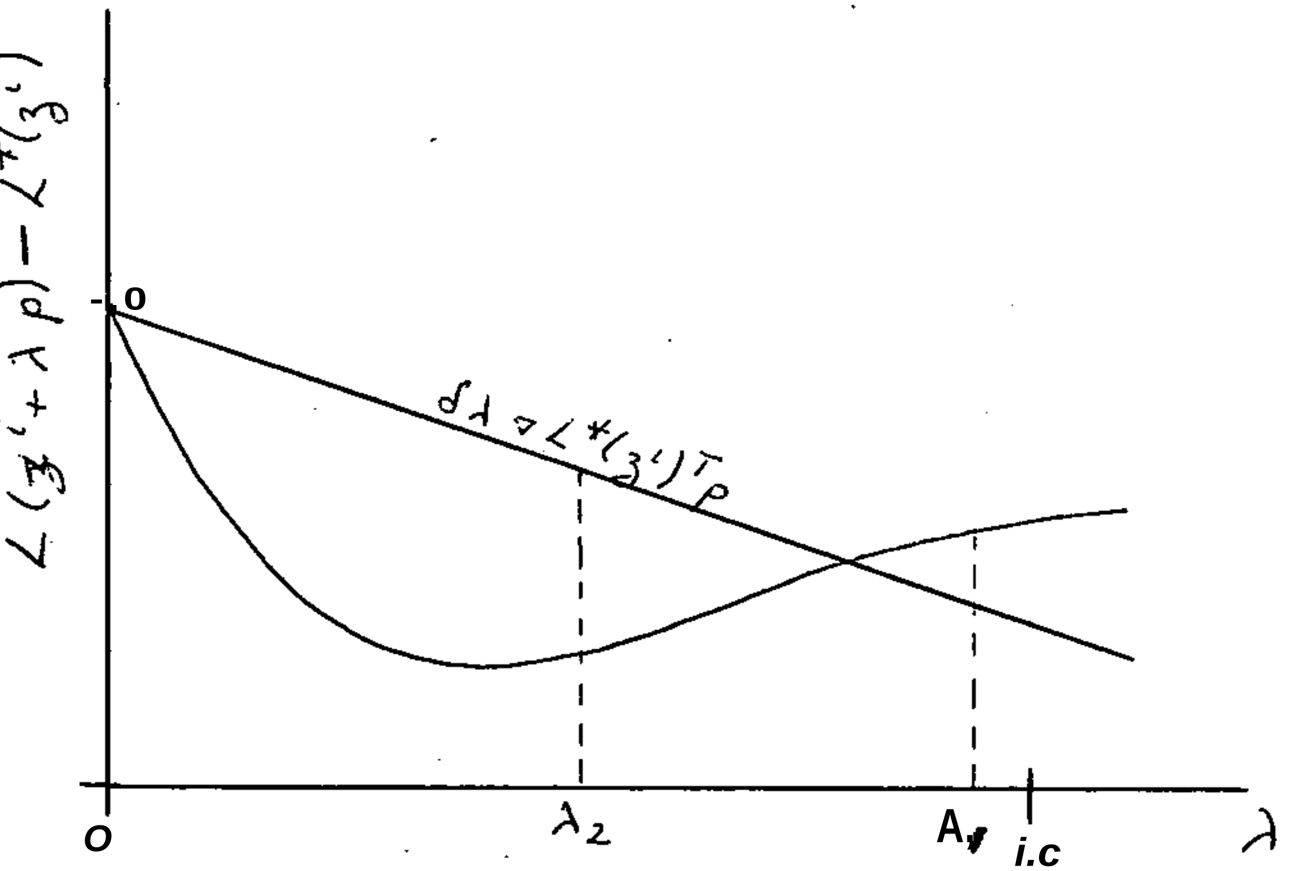
s.t.

$-x_2 + 2x_1^2 - x_1^3 \le 0$

$-x_2 + 2(1-x_1)^2 - (1-x_1)^3 \le 0$

Figure 2

Chamberlain Cycling Problem



Armijo Inequality

$$L^*(z^i+\lambda p) \le L^*(z^i) + \delta\lambda \nabla L^*(z^i)^T p$$

Figure 3

Stepsize Condition

λ_1 - condition not satisfied
λ_2 - condition satisfied

Table 1

Comparison of SQP Algorithms on
15 Test Problems

Function evaluations (CPU msec)

	N	M	MEQ	OPT	OPTHP	WDOG	
Al	2	CM	0	4 (117)	cycles	cycles	
A2	1	CM	0	3 (87)	cycles	cycles	
Bl	5	10	0	5 (373)	5 (360)^"	5 (584)	
B3	5	6	0	3 (197)	3 (198)	4 (433)	
B4	4	0	0	52 (2184)	52 (2256)	54 (1780)	
B6	6	4	4	13 (1201)	25 (1712)	13 (2357)	
C3	2	3	o	10 (33)	10 (34)	10 (212)	
C5	3	2	ot	9 (324)	9 (327)	7 (330)	
C13	5	6	o	4 (230)	4 (244)	4 (537)	
C24	2	2	o	4 (100)	8 (125)	6 (121)	
D4	10	11	en	30 (6865)	30 (6837)	52 (47796)	failed
D5	10	3	en	30 (2935)	30 (2911)	28 (5882)	
D9	4	4	o	5 (197)	5 (190)	5 (396)	
E	2	1	0	10 (333)	10 (309)	10 (177)	
F	4	3	0	12 (766)	14 (761)	11 (678)	

A. Chamberlain (1979)
B. Colville (1968)
C. Himraelblau (1972)
D. Bracken and McCormick (1968)
E. Schuldt (1975)
F. Itosen and Suzuki(1965)

7* bis 2

Old Scaling

[illegible]

[illegible]

Iterations to convergence

	WDLM	OPT
[illegible]		
1a	4(8,82[illegible])	3(4,4[illegible])
1b	3(5,77[illegible])	3(4,549[illegible])
$\eta = 10^{-3}$		
Ridge		
2a	7(8.[illegible]2)	8(7.756)
2b	11([illegible])	18(16,56[illegible])
$\eta = 10^{-3}$		
[illegible]		
[illegible]	20(145,30)	12(76,713)
	(ϕ=1483.17)	(ϕ=1482.91)
$\eta = 10^{-3}$		

Starting points

a − [1.4,30]

b − [0.5,40]

Table 3

New Scaling Results On
Appropriate Test Problems
Using OPT

# function evaluations

	Unsealed	Scaled	
B3	3	3	
B6	13	10	
C3	10	10	(converged to different local min.)
C13	4	4	
D4	30	7	
D9	5	5	
F	12	12	