On Computational Aspects of Bounded Linear Least Squares Problems

ACHIYA DAX
Hydrological Service, Jerusalem

The paper describes numerical experiments with active set methods for solving bounded linear least squares problems It concentrates on two problems that arise in the implementation of the active set method. One problem is the choice of a good starting point. The second problem is how to move out of a "*dead point.*" The paper investigates the use of simple iterative methods to solve these problems. The results of our experiments indicate that the use of Gauss-Seidel iterations to obtain a starting point is likely to provide large gains in efficiency. Another interesting conclusion is that dropping one constraint at a time is advantageous to dropping several constraints at a time.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization—*least squares methods*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Active set methods, bounded least squares, dead points, numerical experiments, starting point

1. INTRODUCTION

This paper considers bounded linear least squares problems of the form

$$\begin{aligned} &\text{minimize} && F(\mathbf{x}) = \frac{1}{2}\| A\mathbf{x} - \mathbf{b} \|^2 \\ &\text{subject to} && \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \end{aligned} \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of unknowns, A is a $m \times n$ matrix and $\mathbf{b} \in \mathbb{R}^m$. The vectors $\mathbf{l}$ and $\mathbf{u}$ belong to $\mathbb{R}^n$ and denote the corresponding lower and upper bounds. A straightforward approach to solve this problem is to use the active set method. (For a general description of this method, see, for examples, Gill et al. [4] or Fletcher [2].) This is an iterative method whose kth iteration starts with $\mathbf{x}^{(k)}$ and ends with $\mathbf{x}^{(k+1)}$ such that $F(\mathbf{x}^{(k+1)}) < F(\mathbf{x}^{(k)})$. In our case the kth iteration is composed of the following four steps.

Step 1. *Compute a search direction.* Let $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ denote the current residuals vector and let $P^{(k)} = diag\{p_1, \ldots, p_n\}$ denote the projection

Author's Address: A. Dax, Hydrological Service, P.O.B. 6381, Jerusalem 91060, Israel.

matrix that corresponds to $\mathbf{x}^{(k)}$. The elements of $P^{(k)}$ are defined by the following rule: If $l_j < x_j^{(k)} < u_j$ then $p_j = 1$, otherwise $p_j = 0$. Then the search direction, $\mathbf{y}$, is obtained by solving the problem

$$\begin{array}{ll} \text{minimize} & \| A\mathbf{y} - \mathbf{r}^{(k)} \|^2 \\ \text{subject to} & P\mathbf{y} = \mathbf{y}. \end{array} \tag{1.2}$$

Step 2. *Compute a step length.* If $F(\mathbf{x}^{(k)} + \mathbf{y}) = F(\mathbf{x}^{(k)})$ then skip to Step 3. Otherwise set

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda \mathbf{y} \tag{1.3}$$

where λ is the largest number in the interval $[0, 1]$ that keeps $\mathbf{x}^{(k+1)}$ feasible. Then start the next iteration. (If $\lambda = 1$ then the next iteration should start in Step 3.)

Step 3. *Test for optimality.* Let the vector $\mathbf{g}^{(k)} = (g_1^{(k)}, \ldots, g_n^{(k)})^{\mathrm{T}}$ denote the gradient of $F(\mathbf{x})$ at $\mathbf{x}^{(k)}$, i.e., $\mathbf{g}^{(k)} = -A^{\mathrm{T}}\mathbf{r}^{(k)}$. Let the vector $\hat{\mathbf{g}} = (\hat{g}_1, \ldots, \hat{g}_n)^{\mathrm{T}}$ be obtained from $\mathbf{g}^{(k)}$ by the following rule: If $x_j^{(k)} = l_j$ and $g_j^{(k)} \geq 0$ set $\hat{g}_j = 0$, if $x_j^{(k)} = u_j$ and $g_j^{(k)} \leq 0$ set $\hat{g}_j = 0$, otherwise set $\hat{g}_j = g_j^{(k)}$. The vector $\hat{\mathbf{g}}$ enables us to test whether the Kuhn–Tucker optimality conditions hold at $\mathbf{x}^{(k)}$. If $\hat{\mathbf{g}} = 0$, then these conditions hold and the algorithm terminates.

Step 4. *Dropping active constraints.* Compute a new point, $\mathbf{x}^{(k+1)}$, such that $F(\mathbf{x}^{(k+1)}) < F(\mathbf{x}^{(k)})$.

The primary problem in the implementation of the above method is the solution of (1.2). This can be done by updating the QR factorization of the matrix that corresponds to "free" variables. (The jth variable is said to be "free" at $\mathbf{x}^{(k)}$ if $l_j < x_j^{(k)} < u_j$, otherwise it is called "bounded".) The factorization scheme is related to the question of how to choose $\mathbf{y}$ when the solution of (1.2) is not unique. These problems are, however, outside of the scope of this paper. The solution of these problems is discussed by Fletcher and Jackson [3], Lawson and Hanson [6], Schittkowski [8] and others.

This paper considers two secondary problems of the above method. The first one is how to choose the initial point, $\mathbf{x}^{(1)}$. The second one is how to implement Step 4. The idea which is tested here is to solve these problems by using a simple iterative scheme such as the Gauss–Siedel method or the Projected-Gradient method. A brief description of these methods is given in the next two sections. The iterations of these methods need no matrix factorization and therefore are likely to need less computational effort. The use of a few simple iterations at the preliminary stage is likely to generate a better starting point for the active set method. This has two potential advantages: The overall computational effort might be reduced while the accuracy of the solution is likely to improve since there is less updating and downdating.

The active set strategies which are investigated by Fletcher and Jackson [3] interchange one active constraint at a time. On the other hand, the use of

simple iterations in Step 4 allows the interchanging of several active constraints at one step. The use of active set strategies that allow the dropping of many constraints was suggested by Goldfarb [5] in the context of convex quadratic programming. Later Lenard [7] compared the two approaches in the context of nonlinear programming with linear constraints. The results of Lenard indicate that in most cases a strategy which keeps the number of active constraints as small as possible is computationally most efficient. This observation suggests the use of similar strategies in the solution of (1.1). The present paper compares, therefore, the relative efficiency of the two approaches.

The main disadvantage in using simple methods is the lack of the finite termination property and the fact that the method may converge very slowly. Thus the main problem of incorporating simple iterations into the active set method is to decide at what stage one should stop the simple iterations. This problem is discussed in Section 4 where a simple switching rule is derived.

The rest of the paper attempts to answer the question of how the above ideas work in practice. For this purpose, we have conducted extensive numerical experiments with several variants of the active set method. The results of our experiments are quite interesting. The Gauss-Seidel (GS) iterations seem to have a clear advantage over the Projected-Gradient (PG) iterations, while the dropping of several constraints seems to have no advantage over the dropping of one constraint. The idea of using GS iterations to generate a better starting point has been proved very useful. It is shown that in many cases this technique results in a dramatic reduction of the computational effort.

2. THE PROJECTED-GRADIENT METHOD FOR SOLVING (1.1)

The kth iteration of this method, $k = 1, 2, 3, \ldots$, goes as follows. Compute $\hat{\mathbf{g}}$ as in Step 3 of the active set method. If $\hat{\mathbf{g}} = \mathbf{0}$ then $\mathbf{x}^{(k)}$ solves (1.1) and the algorithm terminates. Otherwise set

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda \hat{\mathbf{g}} \tag{2.1}$$

where λ is the largest number in the interval $[0, \|\hat{\mathbf{g}}\|^2 / \|A\hat{\mathbf{g}}\|^2]$ that keeps $\mathbf{x}^{(k+1)}$ feasible.

3. THE GAUSS – SEIDEL METHOD FOR SOLVING (1.1)

The basic iteration of this method is composed of n steps. At the jth step, $j = 1, \ldots, n$, x_j alone is changed in an attempt to reduce the value of $F(\mathbf{x})$ as much as possible. Let $\mathbf{x} = (x_1, \ldots, x_n)^T$ denote the current point and let $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ denote the corresponding residuals vector. Then $\tilde{x}_j$, the new value of x_j, is defined as

$$x_j - \lambda \mathbf{a}_j^T \mathbf{r} / \|\mathbf{a}_j\|^2 \tag{3.1}$$

where $\mathbf{a}_j$ denotes the jth column of A and λ is the largest number in the interval $[0, 1]$ that keeps $\tilde{x}_j$ feasible.

4. A SWITCHING RULE

The motivation behind the use of simple iterations is based on two assumptions. One assumption is that simple iterations need less computational effort than active set iterations. The second assumption is that simple iterations are helpful in identifying the "final" active set. In other words, the simple iterations are an efficient tool for reaching a point $\mathbf{x}^{(k)}$ that has the same free variables as the solution point. Hence the use of simple iterations may save unnecessary updating and downdating of the factorization scheme and, consequently, improve the accuracy of the computed solution. On the other hand, once the final active set is found, there is no advantage in continuing the simple iterations since the active set method reaches the minimizer of (1.1) in one iteration. The last observation forms the basis of the proposed switching rule: The switch takes place as soon as the number of free variables remains unchanged during a fixed number, say μ, of simple iterations. In addition, it is reasonable to bound the number of successive simple iterations by n. Of course, the value of μ which is used in the computation of $\mathbf{x}^{(1)}$ should be larger than that which is used in Step 4. The algorithms which are described in the next section use $\mu = 4$ for the first purpose and $\mu = 2$ for the second one.

5. THE TESTED ALGORITHMS

The effects of simple iterations were studied by testing the following variations of the active set method.

Algorithm A. This algorithm uses PG iterations to compute $\mathbf{x}^{(1)}$ in the preliminary stage and to compute $\mathbf{x}^{(k+1)}$ in Step 4.

Algorithm B. This algorithm uses GS iterations to compute $\mathbf{x}^{(1)}$ in the preliminary stage and to compute $\mathbf{x}^{(k+1)}$ in Step 4.

Algorithm C. Here, no simple iterations are used to generate a better starting point, while Step 4 is carried out with $-\hat{\mathbf{g}}$ as a search direction, that is, by using one PG iteration. (Consequently, if $\mathbf{x}^{(k)}$ is a vertex then $\mathbf{x}^{(k+1)}$ is obtained by a PG iteration.)

Algorithm D. This algorithm is the same as C, except that here Step 4 is carried out by one GS iteration.

Algorithm E. This algorithm does not use simple iterations. The computation of $\mathbf{x}^{(k+1)}$ in Step 4 is done by using $-sign(\hat{g}_s)\mathbf{e}_s$ as a search direction, where

$$|\hat{g}_s| = \max\{|\hat{g}_j|, j = 1, \ldots, n\}$$

and $\mathbf{e}_s$ denotes the sth column of $n \times n$ unit matrix.

Algorithm F. This algorithm uses GS iterations to compute $\mathbf{x}^{(1)}$ in the preliminary stage, while Step 4 is carried out as in E.

Algorithm G. This algorithm starts by calculating $\mathbf{x}^*$, a solution of the unconstrained least squares problem. Then $\mathbf{x}^{(1)}$ is obtained from $\mathbf{x}^*$ by "pruning" the components of $\mathbf{x}^*$ that violate their bound. Step 4 is carried out as in E.

6. THE TEST PROBLEMS

We have used in our experiments four types of test problems.

Random Test Problems

This type of test problem has the form (1.1). The elements of A are random numbers from the interval $[-1, 1]$. The vector $\mathbf{b}$ is defined as $\mathbf{b} = A\tilde{\mathbf{x}}$ where $\tilde{\mathbf{x}}$ is a n-vector whose elements are random numbers from the interval $[-\gamma, \gamma]$. (The random number generator is of uniform distribution.) The lower bounds are $\mathbf{l} = -\mathbf{e}$ and the upper bounds are $\mathbf{u} = \mathbf{e}$ where $\mathbf{e}^T = (1, 1, \ldots, 1)$. The role of γ is to control the ratio between free and bounded variables at the solution point. Clearly, γ should be greater than 1, and as its value increases the number of free variables reduces. We have tested this problem using the starting points $\mathbf{x}^{(0)} = \mathbf{0} = (0, 0, \ldots, 0)^T$ and $\mathbf{x}^{(0)} = \mathbf{e}$.

Hilbert Test Problems

This type also has the form (1.1). Here $a_{ij} = 1/(i + j - 1)$ while $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$ and $\mathbf{x}^{(0)}$ are defined as in Random test problems.

Random and Flat Polytopes

The "polytope" test problems have the form

$$\begin{array}{ll} \text{minimize} & \| \tilde{A}\mathbf{x} - \tilde{\mathbf{e}}_1 \|^2 \\ \text{subject to} & \mathbf{x} \geq \mathbf{0} \end{array} \tag{5.1}$$

where $\tilde{A}$ is a $(m + 1) \times n$ matrix whose first row is $\mathbf{e}^T = (1, 1, \ldots, 1)$ and $\tilde{\mathbf{e}}_1$ is the first column of the $(m + 1) \times (m + 1)$ unit matrix.

Let A denote the $m \times n$ matrix which is composed of the last m rows of $\tilde{A}$. Then the set

$$\{\mathbf{y} \mid \mathbf{y} = A\mathbf{x}, \mathbf{x} \geq \mathbf{0} \quad \textit{and} \quad \mathbf{e}^T\mathbf{x} = 1\}$$

is a polytope in $\mathbb{R}^m$. Let $\mathbf{y}^*$ denote the point on this polytope that has the smallest Euclidean norm. Then it is possible to obtain $\mathbf{y}^*$ by solving (5.1) (see Dax, [1]). The "*Random*" and the "*Flat*" polytope problems were designed by Wolfe [9] to test an algorithm for computing $\mathbf{y}^*$. The difference between the two types of problems lies in the definition of A.

In *Random* polytope problems the elements of A are defined as follows: Let $\mathbf{y} \in \mathbb{R}^m$ be a vector whose elements are random numbers from the interval $[-2, 2]$. Then the elements in the ith row of A, $i = 1, \ldots, m$, are random numbers from the interval $[y_i - 1, y_i + 1]$.

In *Flat* polytopes the elements in the first row of A are random numbers from the interval $[0.995, 1.005]$ while the other elements of A are random numbers from the interval $[-1, 1]$.

Table I. Random test problems with $n = 20$ and $\gamma = 2$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	15.4	10.0	0.9	0.9	0.0	8.8	14.0	13.1	3.5	2.9	0.0	10.7	0.0
Active set iterations	4.4	3.7	15.9	16.0	16.1	3.7	7.0	4.4	15.9	9.9	15.8	4.9	15.8
Dropping iterations	0.1	0.4	0.9	0.9	1.3	0.4	0.7	0.7	3.5	2.9	16 2	1.1	3.1
Free variables	8.0	7.8	7.5	7.5	7.5	7.8	8.8	8.8	8.4	8.4	7.9	8.4	7.4
$m = 20$													
Simple iterations	14.8	8.1	0.2	0.2	0.0	7.8	14.9	8.2	3.0	2.3	0.0	8.2	0.0
Active set iterations	2.2	1.6	11.8	11.8	11.9	1.6	1.4	1.2	10.9	7.0	13.6	1.2	9.7
Dropping iterations	0.1	0.1	0.2	0.2	0.3	0.1	0.0	0.0	3.0	2.3	15.6	0.0	2.4
Free variables	9.7	9.7	9.7	9.7	9.7	9.7	8.7	8.7	8.7	8.7	8.7	8.7	9.7
$m = 30$													
Simple iterations	15.1	7.3	0.2	0.2	0.0	7.3	13.8	7.6	2.7	2.0	0.0	7.6	0.0
Active set iterations	1.2	1.0	11.9	11.9	12.0	1.0	1.2	1.0	9.5	6.3	13.9	1.0	8.5
Dropping iterations	0.0	0.0	0.2	0.2	0.3	0.0	0.1	0.0	2.7	2.0	16.2	0.0	1.1
Free variables	9.6	9.6	9.6	9.6	9.6	9.6	9.1	9.1	9.1	9.1	9.1	9.1	9.6

We have tested these problems on two starting points: $\mathbf{x}^{(0)} = \mathbf{0}$ and $\mathbf{x}^{(0)} = \mathbf{e}/n = (1/n, 1/n, \ldots, 1/n)^T$.

7. NUMERICAL RESULTS

The performance of the tested algorithms was measured by recording the following data:

(1) The number of simple iterations that were executed in the preliminary stage and in Step 4.
(2) The number of active set iterations, i.e., iterations at which problem (1.2) was solved.
(3) The number of iterations at which Step 4 was executed ("dropping iterations").
(4) The number of free variables at the solution point. (This information gives us a further insight into the nature of the test problems.)

The results of our experiments are presented in Tables I–VIII. The values of n, m, γ and $\mathbf{x}^{(0)}$ were chosen to give a wide spectrum of test problems. For each combination of these parameters we have generated ten different test problems. The figures in these tables are, therefore, average numbers that have been obtained by solving ten different problems of the same type, size and starting point. The stopping condition which was used in these experiments is

$$\|\hat{\mathbf{g}}\| \leq 10^{-8}.$$

Table II. Random test problems with $n = 20$ and $\gamma = 5$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	19.5	6.9	1.4	1.1	0.0	6.9	13.6	7.4	2.8	2.4	0.0	7.4	0.0
Active set iterations	1.3	1.1	20.8	20.4	21.0	1.1	2.0	1.4	12.8	5.3	7.2	1.4	18.2
Dropping iterations	0.0	0.0	1.4	1.1	2.2	0.0	0.0	0 0	2.8	2.4	13.6	0.0	1.8
Free variables	3.0	3.7	3.7	3.7	3.7	3.7	4.7	4.7	4.7	4.7	4.7	4 7	3.1
$m = 20$													
Simple iterations	20.0	5.8	0.4	0.4	0.0	5.8	13 3	6.3	2.1	1.9	0.0	6.3	0.0
Active set iterations	0.9	0.9	17 7	17.7	17.7	0.9	0.9	0 8	10.9	3.2	4.3	0.8	16 0
Dropping iterations	0.0	0.0	0.4	0.4	0.4	0.0	0.0	0.0	2.1	1.9	12.7	0.0	0.4
Free variables	4.0	4.0	4.0	4.0	4.0	4.0	4.4	4.4	4.4	4.4	4.4	4.4	2 9
$m = 30$													
Simple iterations	20.4	6.2	0 2	0.2	0.0	6.2	14.9	5.9	1.6	1.7	0.0	5.9	0.0
Active set iterations	1.0	0.8	17.8	17.8	17.8	0.8	1.0	1.0	11.0	3.1	4.7	1.0	14.4
Dropping iterations	0.0	0.0	0.2	0.2	0 2	0.0	0.0	0.0	1.6	1.7	13.5	0 0	0.4
Free variables	3.5	3.5	3.5	3 5	3.5	3.5	3.8	3 8	3.8	3.8	3 8	3.8	3.9

Table III. Hilbert test problems with $n = 20$ and $\gamma = 2$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	54.2	28.6	7.3	7.8	0.0	6.4	57.1	33.5	9.3	10.5	0.0	7.2	0.0
Active set iterations	79.2	57.8	64.9	60.1	28 3	28.9	76 0	86.5	77.6	100 0	21 7	26.2	28.7
Dropping iterations	6.6	7.2	7 3	7.8	5.7	6.8	6.5	8.2	9.3	10.5	19.0	7.2	7.1
Free variables	2.8	3.0	2.8	2.6	2.7	2.7	3.5	2.6	2.8	3.5	2.7	2.8	2.7
$m = 20$													
Simple iterations	42.9	26.9	7.5	6 6	0 0	7.2	70 5	42.8	12.7	16.4	0.0	10.5	0.0
Active set iterations	52.5	48.4	65.2	53.6	27.7	27.2	93.4	87.7	99.0	116.9	30.3	33.4	27.8
Dropping iterations	4.3	6.2	7.5	6.6	6.2	6.6	8.0	10 1	12.7	16.4	23.7	9.9	7.5
Free variables	3.5	3.5	3.5	3.5	3.2	3.2	2.9	2 9	2.8	3.7	2.9	2 9	3.0
$m = 30$													
Simple iterations	55.3	27.6	9.6	8.2	0.0	6.4	68.5	35.9	11.2	12.4	0.0	11.1	0.0
Active set iterations	79.3	51.7	85.2	66.3	31.8	28 9	90.2	61.2	89.6	98.1	31.2	31.2	29 2
Dropping iterations	6.6	6 8	9.6	8.2	8.3	7.3	7.7	7.8	11.2	12.4	26.0	10.6	8.0
Free variables	3.9	3.1	3.0	3.0	3.0	2.9	2.9	2.8	2.7	2.7	2.7	2.7	3.1

8. DISCUSSION AND CONCLUSIONS

The comparison of Algorithm A to B and Algorithm C to D indicates that GS iterations have a clear advantage over PG iterations. The reason is, probably, the ability of the GS iterations to add many active constraints at one iteration. This, and the fact that GS iterations need less computational effort, seems to exclude the use of PG iterations.

The comparison of Algorithm E with C and D indicates that there is no gain in using simple iterations at Step 4. In Hilbert test problems, the simple

Table IV. Hilbert test problems with $n = 20$ and $\gamma = 5$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	33.2	17.7	6.3	5.6	0.0	9.1	35.7	14.7	7.4	4.6	0.0	6.8	0.0
Active set iterations	32.3	19.5	53.3	45.7	28.2	13.3	35.3	19.5	59.3	32.9	9.9	8.2	23.2
Dropping iterations	2.9	2.8	6.3	5.6	8.2	3.4	3.1	2.6	7.4	4.6	13.0	3.0	7.8
Free variables	1.2	1.2	1.2	1.2	1.2	1.2	0.8	0.8	0.8	0.8	0.8	0.8	0.8
$m = 20$													
Simple iterations	34.2	21.3	6.2	5.7	0.0	11.7	33.2	24.4	8.9	10.3	0.0	11.6	0.0
Active set iterations	33.7	22.9	57.5	44.7	28.6	11.1	34.1	30.4	67.0	78.5	14.1	12.8	24.7
Dropping iterations	2.7	3.0	6.2	5.7	8.5	2.6	3.0	4.3	8.9	10.3	16.9	3.8	8.3
Free variables	1.3	1.3	1.3	1.3	1.3	1.3	1.2	1.2	1.2	1.2	1.2	1.2	1.4
$m = 30$													
Simple iterations	44.7	23.4	5.6	5.3	0.0	7.4	35.5	23.1	10.1	7.9	0.0	9.6	0.0
Active set iterations	52.4	38.3	53.2	45.3	27.4	19.0	40.7	35.8	75.7	62.3	12.9	15.8	26.4
Dropping iterations	4.4	4.9	5.6	5.3	7.2	5.1	3.8	4.2	10.1	7.9	16.2	3.9	9.6
Free variables	1.8	1.8	1.8	1.8	1.7	1.7	1.5	1.5	1.5	1.5	1.5	1.5	1.5

Table V. Random Polytope test problems with $n = 20$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	20.3	9.5	1.2	1.7	0.0	9.5	19.8	9.8	0.5	0.5	0.0	9.5	0.0
Active set iterations	2.7	1.4	18.3	3.6	3.4	1.4	2.5	2.0	18.5	18.5	18.5	2.0	6.3
Dropping iterations	0.0	0.0	1.2	1.7	4.1	0.0	0.0	0.1	0.5	0.5	0.5	0.1	3.5
Free variables	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5	3.5
$m = 20$													
Simple iterations	18.6	11.1	1.2	2.2	0.0	10.2	19.9	12.1	0.1	0.1	0.0	11.1	0.0
Active set iterations	3.7	1.9	16.7	4.8	5.0	1.9	2.7	2.3	17.2	17.2	17.2	2.4	7.9
Dropping iterations	0.0	0.3	1.2	2.2	5.4	0.3	0.0	0.3	0.1	0.1	0.1	0.4	4.1
Free variables	4.7	4.7	4.7	4.7	4.7	4.7	4.0	4.0	4.0	4.0	4.0	4.0	4.7
$m = 30$													
Simple iterations	18.6	10.2	1.1	2.3	0.0	9.3	17.5	12.8	0.0	0.0	0.0	12.2	0.0
Active set iterations	3.3	2.0	16.0	5.2	5.3	2.0	2.6	2.3	15.5	15.5	15.5	2.3	8.9
Dropping iterations	0.0	0.3	1.1	2.3	5.8	0.3	0.0	0.2	0.0	0.0	0.0	0.2	4.7
Free variables	5.2	5.2	5.2	5.2	5.2	5.2	5.5	5.5	5.5	5.5	5.5	5.5	5.2

iterations free almost all the bounded variables and this doubles the amount of work. A further disadvantage of using simple iterations at Step 4 lies in the complication of the matrix factorization scheme. Hence the traditional strategy of removing one constraint at a time seems to be a superior approach.

On the other hand, the use of GS iterations to provide a better starting point seems to be a useful idea. For well-conditioned problems, such as

Table VI. Flat Polytope test problems with $n = 20$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	13.0	16.0	3.3	5.3	0.0	11.8	12.2	14.9	2.6	2.7	0.0	8.6	0.0
Active set iterations	16.8	8.2	19.8	14.8	15.1	8.8	16.3	10.3	21.7	21 2	20.5	10.0	12.3
Dropping iterations	1.6	1.3	3.3	5.3	12.9	2.1	1.5	1.8	2.6	2.7	4.5	2.4	7.2
Free variables	9.7	9.7	9.7	9.7	9.7	9.7	9.5	9.5	9.5	9.5	9.5	9.5	9.7
$m = 20$													
Simple iterations	7.5	10.3	1.4	3.0	0.0	9.7	10.1	10 5	0.8	0.8	0.0	9.6	0.0
Active set iterations	5.6	3.1	8.4	7.7	15.7	3.0	4.7	2.7	8.7	8.7	8.7	2.7	6.1
Dropping iterations	0.0	0.2	1.4	3.0	15.1	0.2	0.2	0.3	0.8	0.8	0.8	0.3	3.9
Free variables	13.5	13.5	13 5	13.5	13.5	13.5	13.9	13.9	13.9	13.9	13.9	13 9	13.5
$m = 30$													
Simple iterations	6.6	10.1	1.0	2.8	0.0	9.9	9.7	9.8	0.0	0.0	0.0	9.8	0.0
Active set iterations	4.1	1.8	5 4	6.4	17.0	1.9	2.4	1.7	5.6	5.6	5.6	1 7	7.3
Dropping iterations	0.0	0.1	1.0	2.8	16.8	0.1	0.0	0.0	0.0	0.0	0.0	0.0	4.1
Free variables	15.6	15.6	15.6	15.6	15.6	15.6	15.4	15.4	15.4	15.4	15.4	15.4	15.6

Table VII. Random Polytope test problems with $n = 40$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	38.8	10.2	1.5	2.1	0.0	9.3	39.3	12.4	0.7	0 6	0.0	11.5	0.0
Active set iterations	3.2	2.7	38.9	6.4	4.2	2.7	2.5	1.8	39.1	38.8	39.0	1.8	7.8
Dropping iterations	0.0	0.3	1.5	2.1	4.5	0.3	0.0	0.3	0.7	0.6	0.9	0.3	4.2
Free variables	3.7	3.7	3.7	3.7	3.7	3.7	3.8	3.8	3.8	3 8	3.8	3.8	4.1
$m = 20$													
Simple iterations	38.0	14.0	1.2	2.1	0.0	13.4	38.7	13.0	0.5	0.5	0.0	12.7	0.0
Active set iterations	3.3	2.2	36.5	6.9	5.9	2.2	3.6	2.0	37.7	37.6	37.8	2.0	8.3
Dropping Iterations	0.0	0.2	1.2	2.1	6.0	0.2	0.0	0.1	0.5	0.5	0 6	0.1	4.3
Free variables	5.0	5 0	5.0	5.0	5 0	5.0	4 4	4.4	4.4	4.4	4 4	4.4	4 4
$m = 30$													
Simple iterations	39.6	13.4	1.3	2.7	0.0	12.2	36.5	15.3	0.4	0.4	0.0	14.1	0.0
Active set iterations	3.4	2.3	36.1	7.3	6.4	2.3	3.9	2.0	35.3	35.3	35.4	2.0	10.1
Dropping iterations	0.1	0.4	1.3	2.7	6.7	0.4	0.0	0.4	0.4	0.4	0.5	0.4	5.4
Free variables	5.7	5.7	5.7	5.7	5.7	5.7	6.6	6.6	6.6	6.6	6.6	6.6	5.8

Random test problems or Polytope test problems, Algorithm F has a clear advantage over Algorithm E. In Hilbert test problems the GS iterations are less successful because of their slow rate of convergence. Another exception occurs in cases when only few interchanges of active constraints are needed to reach the solution point. Nevertheless, inspection of these exceptions shows that even in these cases the use of GS iterations does not cause a significant waste of efforts. It is possible therefore to conclude that the use of

Table VIII. Flat polytope test problems with $n = 40$

	Starting Point												
	$\mathbf{x}^{(0)} = \mathbf{0}$						$\mathbf{x}^{(0)} = \mathbf{e}$						$\mathbf{x}^{(0)} = \mathbf{x}^*$
	A	B	C	D	E	F	A	B	C	D	E	F	G
$m = 10$													
Simple iterations	20.9	25.4	3.7	7.1	0.0	16.6	8.7	23.4	2.7	2.7	0.0	14.5	0.0
Active set iterations	41.8	24.4	50.9	39.0	27.0	22.7	36.1	26.4	54.7	47.3	43.0	24.9	26.7
Dropping iterations	2.3	2.5	3.7	7.1	19.0	4.7	1.1	2.6	2.7	2.7	6.0	3.8	13.7
Free variables	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	9.9
$m = 20$													
Simple iterations	26.6	26.6	5.5	6.5	0.0	14.3	32.2	22.1	4 6	4.5	0.0	13.4	0.0
Active set iterations	31.6	25.7	49.3	30.4	36.4	22.5	26.9	20.0	52.6	47.6	46.1	18.0	27.4
Dropping iterations	3.1	3.6	5.5	6.5	28.2	7.2	2.6	2.7	4.6	4.5	12.1	4.9	15.9
Free variables	19.0	19.0	19.0	19.0	19.0	19.0	19.1	19.1	19.1	19.1	19.1	19.1	19.2
$m = 30$													
Simple iterations	18.2	14.2	3.9	3.7	0.0	12.6	20.4	16.7	3.1	2.8	0.0	14.6	0.0
Active set iterations	14.5	5.6	31.1	14.6	30.1	5.9	10.2	4.7	31.9	29.5	36.2	5.1	20.3
Dropping iterations	1.0	0.5	3.9	3.7	27.5	0.8	0.7	0.7	3.1	2.8	9.6	1.1	12.8
Free variables	23.9	23.9	23.9	23.9	23.9	23.9	24.0	24.0	24.0	24.0	24.0	24.0	23.7

GS iterations to provide a better starting point is likely to cause a significant saving of computational effort.

REFERENCES

1. DAX, A. The smallest point of a polytope. *J. Optimization Theory Appl. 64* (1990), 419–422.
2. FLETCHER, R. *Practical Methods of Optimization. Vol. 2, Constrained optimization.* Wiley, New York, 1981.
3. FLETCHER, R., AND JACKSON, M. P. Minimization of a quadratic function of many variables subject only to upper and lower bounds. *J. Inst. Math. Appl. 14* (1974), 159–174.
4. GILL, P. E., MURRAY, W., AND WRIGHT, M. H. *Practical Optimization.* Academic Press, London, 1981.
5. GOLDFARB, D. Extensions of Newton's method and simplex methods for solving quadratic programs. In *Numerical Methods for Nonlinear Optimization*, F. A. Lootsma, Ed. Academic Press, 1972, pp. 239–254.
6. LAWSON, C. L., AND HANSON, R. J. *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, N.J., 1974.
7. LENARD, M. L. A computational study of active set strategies in nonlinear programming with linear constraints. *Math. Program. 16* (1979), 81–97.
8. SCHITTKOWSKI, K. The numerical solution of constraints linear least-squares problems. *IMA J. Numer. Anal. 3* (1983), 11–36.
9. WOLFE, P. Algorithm for a least-distance programming problem. *Math. Program. Stud. 1* (1974), 190–205

Received June 1988; revised January 1989; accepted January 1990