SOLVING LINEAR INEQUALITIES IN A LEAST SQUARES SENSE

R. BRAMLEY AND B. WINNICKA *

Abstract. In 1980, Han [6] described a finitely terminating algorithm for solving a system $Ax \leq b$ of linear inequalities in a least squares sense. The algorithm uses a singular value decomposition of a submatrix of A on each iteration, making it impractical for all but the smallest problems. This paper shows that a modification of Han's algorithm allows the iterates to be computed using QR factorization with column pivoting, which significantly reduces the computational cost and allows efficient updating/downdating techniques to be used. The effectiveness of this modification is demonstrated, implementation details are given, and the behaviour of the algorithm discussed. Theoretical and numerical results are shown from the application of the algorithm to linear separability problems.

Key words. iterative methods, linear inequalities, least squares, linear separability

AMS(MOS) subject classifications. 65F10, 65F20, 65F30, 65K05

1. Introduction. Let $A \in \Re^{m \times n}$ be an arbitrary real matrix, and let $b \in \Re^m$ a given vector. A familiar problem in computational linear algebra is to solve the system $Ax = b$ in a least squares sense; that is, to find an x^* minimizing $||Ax - b||$. Here and throughout this paper, $\| \cdot \|$ refers to the two-norm. Such an x^* solves the *normal equations* $A^T(Ax - b) = 0$, and the optimal residual $r^* = b - Ax^*$ is unique (although x^* need not be). The least squares problem is usually interpreted as corresponding to multiple observations, represented by the rows of A and b, on a vector of data x. The observations may be inconsistent, and in this case a solution is sought that minimizes the norm of the residuals. More information about linear least squares problems and solution techniques can be found in [9, 14, 4].

A less familiar problem to numerical linear algebraists is to solve systems of linear *inequalities* $Ax \leq b$ in a least squares sense, but the motivation is similar: if a set of observations places upper or lower bounds on linear combinations of variables, we want to find x^* minimizing $||(Ax - b)_+||$, where the i^{th} component of the vector v_+ is the maximum of zero and the i^{th} component of v. However, potential applications extend beyond simple data analysis, and include linear separability problems. That application requires finding a hyperplane that best separates two point sets; when the two sets are not linearly separable, a hyperplane that correctly separates the largest number of points is desired. Although an $\mathcal{L}_1$ norm formulation using linear programming seems more natural for this problem, the $\mathcal{L}_2$ norm formulation described

* Work supported by NSF grant CDA-9223008

here provides comparable solutions.

When the system $Ax \leq b$ is consistent, that is, when a solution exists that satisfies all the inequalities, then phase I of any standard linear programming method can find it. Futhermore when the system is not consistent, linear programming can identify that case, but does not directly provide an "optimal" solution. Other methods developed for solving linear inequalities include an unusual algorithm by Stewart [13], which defines a function that diverges in a direction that converges to a solution of the inequalities; if no solution exists, the function converges to a unique minimum.

One way of solving the problem is to state it as the quadratic programming problem in (x, z)

$$(\text{QP}) = \begin{cases} \min \frac{1}{2} z^T z \\ \text{subject to } \; Ax - b \leq z \end{cases} \tag{1}$$

However, there are serious numerical difficulties with solving a quadratic programming problem that has a singular objective function; furthermore, most methods require an active set strategy that can be difficult to implement, particularly when it is necessary to decide which entries to drop from the active set. The analogue of an active set for the algorithm described in this paper is automatically determined without difficult decisions of when to drop a constraint. In particular, the *numerical* determination of the active set consists of a test against zero, without the need to introduce machine or problem dependent tolerances.

The only algorithm specifically designed for solving arbitrary systems of linear inequalities in a least squares sense was developed by S.–P. Han [6]. That algorithm requires finding the minimum norm least squares (equality) solution to systems $A_I x = b_I$, where A_I is a submatrix of A consisting of some rows of A. This implies that a singular value decomposition or a complete orthogonal decomposition of A_I is required on every iteration. Both of these decompositions are relatively expensive to compute, and there currently are no effective update/downdate methods that allow the reuse of work performed on a previous iteration. This paper will show that a minor modification of Han's algorithm allows an implementation using a QR factorization with column pivoting instead, and both the robustness and finite termination of Han's algorithm are retained.

Section 2 of this paper defines notation and reviews some basic properties of least squares solutions for linear inequalities, most of which can be found in [6]. Section 3 outlines Han's algorithm and Section 4 presents and validates the minor change in the convergence proof that allows QR with column pivoting to be used. Section 5 gives implementation details and testing results. As an illustration of how the algorithm can be used in an application area, Sections 6 and 7 examine the linear separability problem. Linear separability problems start with two sets of experimental data points,

where the points in one set have a certain property and points in the other do not. A hyperplane that best separates the two point sets is found and then used to classify future data points as having or not having the relevant property.

2. Basics of systems of linear inequalities. This Section summarizes some fundamental properties of linear inequalities from Han's technical report, and proofs of the results can be found in [6]. Let $A \in \Re^{m\times n}$ be an arbitrary real matrix, and let $b \in \Re^m$ be a given vector. No relation is assumed between m and n, and the matrix A can be rank-deficient, ill-conditioned, or even the zero matrix. Let the rows of A be denoted a_i^T, $i = 1, \ldots, m$. We want to find an $x \in \Re^n$ solving the system

$$Ax \leq b \tag{2}$$

in some sense. System (2) is interpreted componentwise, so we want $a_i^T x \leq b_i$ for all $i = 1, \ldots, m$. Possibly no such x exists. As an example, consider the system

$$A = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}, b = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

which is equivalent to $x_1 \leq 1$ and $x_1 \geq 2$. In these cases, we want a *least squares* solution, which we now define. Given a vector $v \in \Re^l$, define its positive part as the vector $v_+ \in \Re^l$ with components given by $v_+^i = \max\{0, v^i\}$. A least squares solution to (2) is any vector x that minimizes

$$f(x) = \frac{1}{2}\|(Ax - b)_+\|^2. \tag{3}$$

Analogous to the linear equality case, we can also define the (necessarily unique) x of minimum norm that solves $x = \text{argmin } \|(Ax - b)_+\|$, but the method analyzed here does not provide a minimum norm solution. For example when $m < n$ and A is full–rank, the method finds one of the infinite number of possible solutions, but which one depends on the starting point. Also, the function $f(x)$ is convex, continuously differentiable, and piecewise quadratic. Differentiating gives the analogue of the normal equations:

PROPOSITION 2.1. *$x^* \in \Re^n$ solves (2) if and only if $A^T(Ax^* - b)_+ = 0$.*

The proposition follows immediately from the convexity of f and the relation $\nabla f(x) = A^T(Ax - b)_+$.

Keeping in mind the similarity of Proposition 2.1 and the normal equations for equality linear least squares problems, we define the residual vector as $z = (Ax - b)_+$. This residual is zero if and only if the system of inequalities is consistent and x is a solution. Furthermore, by noting the equivalence of (2) and the quadratic programming problem (1), which is convex, we have

PROPOSITION 2.2. *For any matrix $A \in \Re^{m\times n}$ and vector $b \in \Re^n$, a least squares solution to (2) exists. The optimal residual vector $z^* = (Ax^* - b)_+$ is unique, and x is a least squares solution if and only if $(Ax - b)_+ = z^*$.*

Finally, we note that the gradient of f is globally Lipschitz of order 1, with a Lipschitz constant of $\|A\|^2$:

PROPOSITION 2.3. $\|\nabla f(x) - \nabla f(y)\| \leq \|A\|^2\|x - y\|$, *for all* $x, y \in \Re^n$.

Again the proof is straightforward, using the relation $\nabla f(x) = A^T(Ax^* - b)_+$. Note however that the Hessian of f fails to exist at points where $a_i^T x = b_i$ for any $i = 1, 2, \ldots, m$.

3. Outline of Han's algorithm. Two notations are needed for the statement of Han's algorithm. First, $G^\dagger$ denotes the pseudo-inverse of the matrix G [12]. In practice, all that is needed is the action of $G^\dagger$ on a vector f, not the linear operator itself in explicit form. The vector $G^\dagger f$ is the minimum norm, least squares solution to the problem of minimizing $||Gy-f||$, and can be computed by a QR factorization when G is full-rank, or by singular value or complete orthogonal decomposition otherwise. See Chapter 5 of [4] for details on computing these and other factorizations for linear least squares.

Second, let $I \subseteq \{1, 2, \ldots, m\}$ be an index set. Then A_I is the submatrix of A consisting of rows with indices in I. With this definition, $A_I \in \Re^{|I|\times n}$, where $|I|$ is the cardinality of I. The vector b_I can be defined similarly. Using this notation, the algorithm is:

ALGORITHM 3.1: Let $A \in \Re^{m\times n}$, $b \in \Re^m$, a starting point $x^0 \in \Re^n$, and $\epsilon > 0$ be given.

Initialize:

- Set $k = 0$ (Iteration number)
- Set $r^0 = b - Ax^0$
- Set $I = \{i : a_i^T x^0 \geq b_i\} = \{i : r_i^0 \leq 0\}$ (Set of active indices)
- Set $\rho^0 = ||r_I^0||$ (initial residual norm)

Iterate: While $(\rho^k > \epsilon)$

- $d^k = A_I^\dagger r_I^k$
- $\lambda = \text{argmin } f(x^k + \lambda d^k)$, where $f(x)$ is defined by equation (3).
- $x^{k+1} = x^k + \lambda d^k$
- $r^{k+1} = b - Ax^{k+1}$
- $I = I_{k+1} = \{i : a_i^T x^{k+1} \geq b_i\}$ (New set of active indices)
- $\rho^{k+1} = ||(r^{k+1})_I||$

- $k = k + 1$

Superscripts in the above algorithm indicate the iteration number. Note that I and λ depend on k also, but for clarity, we omit the k when examining a single step of the algorithm. The exact line search for λ is computationally reasonable, since $\theta(\lambda) = f(x^k + \lambda d^k)$ is piecewise quadratic, convex, and continuous; we can simply search through the knot points to isolate an interval on which $\theta(\lambda)$ is quadratic, then interpolate. Numerical testing shows that the algorithm is in fact sensitive to the line search procedure, and it is worthwhile to consider other methods.

The fundamental result Han established about this algorithm is that it converges in a finite number of steps to some minimizer of (3). The proof relies on the following properties, which are readily verified:

PROPOSITION 3.1. *For any matrix $A \in \Re^{m\times n}$ and vector $b \in \Re^n$, let $f(x)$ be given by (3), $x = x^k$, $d = d^k$, and $I = I(x^k)$. Then*

1. *$\nabla f(x) = -A_I^T A_I d$, so $d^T \nabla f(x) = \|A_I d\|^2$, and d is a descent direction for $f(x)$.*
2. *d is the minimum norm least squares solution to the problem: minimize $\|A_I d - r_I\|$*

4. A more efficient version of Han's algorithm. To make this algorithm practical, d^k needs to be computed by means of a cheaper factorization of A_I. Han's algorithm uses the pseudo-inverse to define d^k in order to limit the growth of the component of x^k in the null space of A_I. As mentioned in the introduction, we now show that this can be relaxed, allowing cheaper computation of d^k. Omitting the superscript k temporarily, let $d_{\text{svd}} = A_I^\dagger r_I$. The general least squares solution to $A_I d = r_I$ is given by $d = d_{\text{svd}} + (I - A_I^\dagger A_I)y$ for $y \in \Re^n$. Note that $(I - A_I^\dagger A_I)y$ is in the null space of A_I, so that $A_I d = A_I d_{\text{svd}}$. This is critical since, with one exception, the proof of finite termination in [6] involves only the quantity $A_I d$, not d. Before addressing that exception, we recall a result of Golub and Pereyra [5]:

THEOREM 4.1. *Let G be an $m \times n$ matrix, and let $GP = Q\hat{R}$ be the QR with column pivoting factorization of G. Partition*

$$\hat{R} = \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}, \tag{4}$$

where R_1 is $r \times r$, upper triangular, and invertible, and R_2 is $r \times n - r$. Let y_{svd} be the minimum norm least squares solution to the problem: minimize $\|Gy - f\|$, let $Q^T f = (c^T, g^T)^T$, where $c \in \Re^r$, and set

$$y = P \begin{pmatrix} R_1^{-1} c \\ 0 \end{pmatrix} \tag{5}$$

Then $\|y\|^2 \leq (1 + \|R_1^{-1}R_2\|^2)\|y_{\text{svd}}\|^2$

The application of this result to the linear inequality algorithm tells us that using QR factorization with column pivoting to compute the search direction d will always provide a d that is bounded by a constant times the norm of d_{svd}, where the constant does not depend on x or the iteration number k:

THEOREM 4.2. *Let d be computed in Han's algorithm using QR factorization with column pivoting. Then there is $C_R \geq 0$ such that $\|d\| \leq C_R\|d_{\text{svd}}\|$ independently of the iteration number k.*

Proof. For a given index set I, let R_1 and R_2 be the factors defined by applying Theorem 4.1 to the problem: miminize $\|A_I d - r_I\|$. Let $C_I = \sqrt{(1 + \|R_1^{-1}R_2\|^2)}$, and let $C_R = \max\{C_I\}$, where the maximum is taken over all possible index sets I. Since the number of such index sets is finite, C_R is well-defined. The result follows immediately from the last Theorem. □

We now prove the main convergence result needed:

THEOREM 4.3. *Let the sequences of vectors x^k and d^k be computed using Algorithm 3.1, with d^k computed using QR factorization with column pivoting on A_I. Then either the algorithm stops after a finite number of iterations, or*

$$\lim_{k\to\infty} \nabla f(x^k) = 0 \tag{6}$$

Proof. If the algorithm stops after a finite number of iterations there is nothing further to prove. Suppose it takes an infinite number of iterations. Then for all steps, A_I is a nonempty matrix, since $I = \phi$ if and only if x^k is a solution. Initially we drop the superscripts k, and consider one step of the algorithm. Since ∇f is continuous and globally Lipschitz with a Lipschitz constant of $\|A\|^2$, from the Mean Value Theorem [11, c.f. Theorem 8.3.1]

$$\begin{aligned} f(x+\lambda d) - f(x) &= \lambda \int_0^1 d^T \nabla f(x + t\lambda d) dt \\ &= \lambda \int_0^1 d^T [\nabla f(x + t\lambda d) - \nabla f(x)] dt + \lambda d^T \nabla f(x) \\ &\leq \lambda \int_0^1 \|d\| \|\nabla f(x + t\lambda d) - \nabla f(x)\| dt + \lambda d^T \nabla f(x) \\ &\leq \lambda \|A\|^2 \int_0^1 \lambda \|d\|^2 t dt + \lambda d^T \nabla f(x), \end{aligned} \tag{7}$$

and so

$$f(x + \lambda d) \leq f(x) + \lambda d^T \nabla f(x) + \frac{\|A\|^2}{2} (\lambda \|d\|)^2 . \tag{8}$$

As a function of λ, the right hand side of the above bound has a minimum at

$$\hat{\lambda} = \frac{-d^T \nabla f(x)}{\|A\|^2 \cdot \|d\|^2}. \tag{9}$$

Substituting this value of λ in (8) and using $d^T \nabla f(x) = -d^T A_I^T A_I d$, we get

$$f(x + \hat{\lambda} d) - f(x) \leq -\frac{1}{2} \left[\frac{d^T \nabla f(x)}{\|A\| \cdot \|d\|} \right]^2 \leq -\frac{1}{2} \left[\frac{\|A_I d\|^2}{\|A\| \cdot \|d\|} \right]^2 \tag{10}$$

Note that if the component of d that lies in the null space of A_I becomes arbitrarily large, the upper bound above can go to zero. However, from Theorem 4.2 $\|d\| \leq C_R \|d_{\text{svd}}\|$ where C_R is independent of the iteration number. Since $d_{\text{svd}} \in \text{range}\ (A_I^T)$, $d_{\text{svd}} = A_I^\dagger A_I d$ and so

$$\|d\| \leq C_R \|A_I^\dagger A_I d_{\text{svd}}\| \leq C_R \|A_I^\dagger\| \cdot \|A_I d_{\text{svd}}\| \leq C \|A_I d_{\text{svd}}\|, \tag{11}$$

where $C = \left(\max_I \|A_I^\dagger\| \right) C_R$. Substituting this bound on d into (10) gives

$$f(x + \hat{\lambda} d) - f(x) \leq -\frac{1}{2C\|A\|^2} \|A_I d\|^2. \tag{12}$$

Since the stepsize λ is chosen by an exact line search,

$$f(x) - f(x + \lambda d) \geq f(x) - f(x + \hat{\lambda} d) \geq \frac{1}{2C\|A\|^2} \|A_I d\|^2. \tag{13}$$

The last inequality holds for all iterations, and since $f(x^k)$ is monotone decreasing and bounded below by zero,

$$\sum_{k=0}^{\infty} \left[f(x^k) - f(x^k + \lambda d^k) \right] \geq \frac{1}{2C\|A\|^2} \sum_{k=0}^{\infty} \|A_{I^k} d_k\|^2 \tag{14}$$

is a finite sum, so $A_{I^k} d^k \to 0$ as $k \to \infty$ and hence $\nabla f(x^k) = -A_{I^k}^T A_{I^k} d^k \to 0$ as $k \to \infty$. □ The rest of Han's proofs, which are not reproduced here, still apply to the modified algorithm, since they only rely on $A_I d$, not d. His arguments in particular show that of the two alternatives in Theorem 4.3, the second alternative (equation (6)) cannot occur. It is worthwhile to compare those results with related ones. As early as 1965, Katznelson [8] established finite termination of an algorithm for solving piecewise linear systems of equations that arise in circuits. More recently Li and Swetits [10] have established finite termination for solving systems of the form $\Phi(x) = Q^T[(Ax + b)_+ + (Cx + d)] = 0$ (c.f. the gradient of $f(x)$). In both cases, they relied on the assumption that within each polyhedral set created by the hyperplanes $H_i = \{x : a_i^T x = b_i\}$, the gradient of Φ is nonsingular. This corresponds to the case where $A_I^T A_I$ is nonsingular for all index sets I. The key idea is that

within each polyhedral set the function is quadratic, and so if the k^{th} iterate lands in the polyhedral set containing the (necessarily unique) minimum, Newton's iteration converges in one step. Since there are a finite number of such polyhedral sets, it is a matter of showing that if an infinite number of the iterates lie in a single polyhedral set, they must converge to a point in the set. The unicity of solutions to $\nabla\Phi(x)d = -\Phi$ allows this by assuring that the iterates remain bounded.

Han's method applies a Gauss–Newton approach to the same problem, and restricts the choice of search directions d^k to minimum norm solutions, in order to have zero growth in the null space component of A_{I^k}. The key idea used in this paper is that some growth in that null space is allowed, provided that it is uniformly bounded over all of the polyhedral sets, that is, over all choices of index set I. This result is important because it allows applying the algorithm to large systems, and the recent development of efficient and reliable orthogonal factorization methods for sparse systems allows it to be applied to the kind of systems that frequently arise in applications. Furthermore, methods for computing QR factorization with some form of pivoting on parallel machines allow implementation on modern high performance computers.

5. Numerical Characteristics of the Algorithm. This Section summarizes results of numerical testing, and further details can be found in [2]. The algorithm has been implemented in Matlab[1], with four ways of generating search directions: the singular value decomposition (SVD), a complete orthogonal factorization (COF), QR factorization with column pivoting (QR), and QR factorization with updating and downdating of the factors (UD), which is used only when it is numerically safe.

The implementation defines the active set exactly as given in Algorithm 3.1. By contrast, active set strategies in quadratic programming methods for (1) involve tolerances and can require numerical determination of ranks of submatrices.

The number of iterations required is sensitive to the line search, since stopping short or overshooting a hyperplane boundary gives a different index set. Two line search methods are tested: a search through the knot points for the piecewise quadratic linesearch function followed by quadratic interpolation, and a binary search method.

Table 1 shows the mean ratios of flops and iterations for 200 random problems of orders 80×40, 40×80, and 400×15. The flops computed are the total flops required, not just those in finding the search direction or performing the line search. The third and fourth columns compare using a SVD to using QR with column pivoting, and show that for all problem sizes SVD is significantly more expensive, even though fewer iterations are required by SVD, especially for 40×80 problem size. This makes intu-

[1] The MathWorks, Inc.

itive sense, because in this case many of the submatrices A_I don't have full column rank. Finding the search direction via QR with column pivoting can introduce components from the null space of A_I, components that may have to be removed by later iterations when the index set changes, and which can introduce numerical instability. However, even when requiring 3 times more iterations, the QR–based method takes fewer flops; less than one tenth as many in some cases. The fifth and sixth columns show the ratios when QR with column pivoting is used, but this time comparing the two line search methods. This shows that the two methods are close in cost, but for overdetermined problems binary search is slightly cheaper. Because binary search is easily coded and evaluating the line search function at a given point requires only the computationally inexpensive task of forming matrix–vector products with A, this suggests it is the preferred method.

Prob Size	Statistic	$\frac{\text{SVD}}{\text{QR}}$ Flops	$\frac{\text{SVD}}{\text{QR}}$ Iters	$\frac{\text{QuadInt}}{\text{BinSearch}}$ Flops	$\frac{\text{QuadInt}}{\text{BinSearch}}$ Iters
40×80	Arith Mean	4.45	0.73	0.98	0.99
	Std Dev	1.82	0.25	0.13	0.12
	Geom Mean	4.13	0.69	0.97	0.99
80×40	Arith Mean	6.18	0.95	1.03	1.12
	Std Dev	1.74	0.23	0.26	0.33
	Geom Mean	5.92	0.93	1.00	1.08
400×15	Arith Mean	2.04	1.00	1.00	1.01
	Std Dev	0.26	0.00	0.06	0.89
	Geom Mean	2.02	1.00	1.00	1.00

TABLE 1

Statistics for Comparing Costs of SVD and QR for Search Direction and Quadratic Interpolation and Binary Search for Line Search

Prob Size	Statistic	$\frac{\text{COF}}{\text{UD}}$	$\frac{\text{SVD}}{\text{UD}}$	$\frac{\text{QR}}{\text{UD}}$
40×80	Arith Mean	1.03	4.56	1.00
	Std Dev	0.41	1.81	0.00
	Geom Mean	0.94	4.20	1.00
80×40	Arith Mean	1.37	6.20	1.37
	Std Dev	0.38	1.77	0.25
	Geom Mean	1.32	5.95	1.35

TABLE 2

Statistics for Ratio of Flops for Search Direction Methods

Table 2 compares the number of flops for four search direction methods, COF, SVD, QR and UD. The ratios of the first three to the last one are given, since UD was found to most often be the most efficient method for overdetermined problems. Although the average cost of COF for underdetermined problems is slightly more than for UD, the geometric mean of 0.94 shows that COF is cheaper for a larger of problems. In

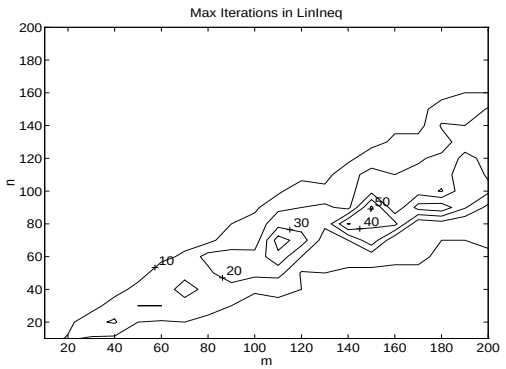

FIG. 1. *Maximum Iterations Required for $m \times n$ Problems*

this case the subproblems tend to have a large null space for A_I, and eliminating that component as COF does significantly reduces the number of iterations. So although on a given iteration COF is more expensive, its overall cost can be reduced.

The conclusions from these experiments are that using a binary line search method is cheaper than a quadratic interpolation as often as not, and altering the algorithm to select the search direction with a QR factorization is an important improvement for the $m \geq n$ case. When $n \gg m$ and the extra freedom introduced in null(A_I) could potentially cause numerical difficulties, COF is preferred.

Figure 1 shows the maximum iterations needed for random problems with varying m and n. In all cases no more than $1 + \max(m, n)$ iterations were needed, and the largest number of iterations occurs when $m \approx 2n$, which may be from having twice as many halfspaces as variables, and so finding the correct index set takes several steps. When $m \gg 2n$, few iterations are needed, since there are generally enough active rows to give an impetus in the directions imposed by rows not currently active, but which are active at the solution. Finally, when $m < n$, only 1–3 iterations are required, since the number of degrees of freedom exceeds the number of "constraints" imposed by the system.

6. Application to the linear separability problem. The linear separability problem is the one of finding a best hyperplane that separates two point sets $\mathcal{A}$ and $\mathcal{B}$ in $\Re^n$. Let $\mathcal{A}$ and $\mathcal{B}$ have m and k points, respectively, and let A and B be matrices with rows giving the coordinates of the points in $\mathcal{A}$ and $\mathcal{B}$. We want to find $w \in \Re^n$ and a scalar γ so that $Aw \leq \gamma e_m$ and $Bw > \gamma e_k$, where $e_i \in \Re^i$ is a vector of all ones.

Clearly not all sets $\mathcal{A}$ and $\mathcal{B}$ can be separated by a hyperplane, so we want to find a hyperplane that is optimal by having fewest points incorrectly classified as belonging

to $\mathcal{A}$ or $\mathcal{B}$. This can be approximated as the least squares inequality problem

$$\begin{aligned} Aw - \gamma e_m &\leq -e_m \\ -Bw + \gamma e_k &\leq -e_k \end{aligned} \tag{15}$$

This formulation specifies that all the points lie outside of the "slab" $H = \{x : \gamma - 1 \leq w^T x \leq \gamma + 1\}$, guarding against the case when all the data points line up on a hyperplane, in which case the least squares problem (15) with zero on the right hand side gives a trivial solution $(w, \gamma) = 0$. In [1], Bennett and Mangasarian formulate this problem in terms of the $\mathcal{L}_1$ norm: minimize

$$\frac{1}{m}\sum_{i=1}^{m}\left(-a_i^T w + \gamma + 1\right)_+ + \frac{1}{k}\sum_{j=1}^{k}\left(b_j^T w - \gamma + 1\right)_+ \tag{16}$$

and then solve it as the linear programming problem: minimize over (w, γ, y, z) the function $(y^T e_m)/m + (z^T e_k)/k$ subject to the constraints $Aw - \gamma e_m + y \geq e_m$, $-Bw + \gamma e_k + z \geq e_k$, and $y, z \geq 0$.

The normalization of terms of the objective function by $1/m$ and $1/k$ assures that nontrivial solutions (w, γ) exist. A similar result holds for the least squares formulation (15):

THEOREM 6.1. *Suppose the two point sets $\mathcal{A}$ and $\mathcal{B}$ have m and k points, respectively, and $m, k > 0$. Then the least squares formulation of (15) has the trivial solution $w = 0$ if and only if*

$$\sum_{i=1}^{m} a_i = \alpha \sum_{j=1}^{k} b_j \tag{17}$$

for some constant α.

Proof. Suppose that the least squares solution of (15) has $w = 0$, let $\tilde{w} = (w^T, \gamma)^T$, and let

$$G = \begin{bmatrix} A & -e_m \\ -B & e_k \end{bmatrix}, \quad g = \begin{pmatrix} -e_m \\ -e_k \end{pmatrix}. \tag{18}$$

Then

$$\begin{aligned} \|(G\tilde{w} - g)_+\|^2 &= \left\| \begin{pmatrix} (1-\gamma)e_m \\ (1+\gamma)e_k \end{pmatrix}_+ \right\|^2 \\ &= \begin{cases} m(1-\gamma)^2, & \text{if } \gamma < -1 \\ k(1+\gamma)^2, & \text{if } \gamma > 1 \\ m(1-\gamma)^2 + k(1+\gamma)^2, & \text{if } -1 \leq \gamma \leq 1 \end{cases} \end{aligned} \tag{19}$$

Furthermore, the active index set I is

$$I = \begin{cases} \{1, 2, \ldots, m\}, & \gamma < -1 \\ \{m+1, m+2, \ldots, m+k\}, & \gamma > 1 \\ \{1, 2, \ldots, m+k\}, & -1 \le \gamma \le 1. \end{cases} \tag{20}$$

From the normal equations, $G^T(G\tilde{w} - g)_+ = 0$. If $\gamma < -1$, this implies

$$0 = \begin{pmatrix} A^T(1-\gamma)e_m \\ -m(1-\gamma) \end{pmatrix} = (1-\gamma) \begin{pmatrix} \sum_{i=1}^m a_i \\ -m \end{pmatrix} \tag{21}$$

and so $m = 0$, a contradiction. Similarly, if $\gamma > 1$, this implies

$$0 = \begin{pmatrix} -B^T(1+\gamma)e_k \\ k(1+\gamma) \end{pmatrix} = (1+\gamma) \begin{pmatrix} -\sum_{j=1}^k b_j \\ k \end{pmatrix} \tag{22}$$

and so $k = 0$, again a contradiction. So $-1 \le \gamma \le 1$, in which case

$$0 = \begin{pmatrix} A^T(1-\gamma)e_m - B^T(1+\gamma)e_k \\ -m(1-\gamma) + k(1+\gamma) \end{pmatrix} = \begin{pmatrix} (1-\gamma)\sum_{i=1}^m a_i - (1+\gamma)\sum_{j=1}^k b_j \\ \gamma(m+k) + (k-m) \end{pmatrix}$$

If $|\gamma| = 1$, the last component implies that one of m or k is zero. So $|\gamma| < 1$, and the theorem holds with $\alpha = (1+\gamma)/(1-\gamma)$. □

It is easy to check condition (17) before beginning computations, and perturbing any entry of G will avoid the trivial solution. By contrast, in the $\mathcal{L}_1$ norm minimizing method of [1], only the objective function for the linear program needs scaling. However, that formulation only guarantees that nontrivial solutions exist, but it does not assure that a linear programming program will find such a nontrivial solution. In any case, the condition in (17) has so far only occured in artificially created problems.

In testing the linear programming and least squares methods, after (w, γ) are found a secondary minimization on γ is performed to improve the solution. The next proposition shows that the inequality least squares problem will translate the separating hyperplane to where the sum of residual violations is balanced; this is not a criterion of the original linear separability problem, and is the reason we follow the iterations with the secondary minimization.

PROPOSITION 6.2. *Let (w, γ) be a least squares solution to the system (15). The sum of residuals corresponding to point set $\mathcal{A}$ equals the sum of residuals corresponding to point set $\mathcal{B}$.*

Proof. From the normal equations for (15),

$$\begin{bmatrix} A^T & -B^T \\ -e_m^T & e_k^T \end{bmatrix} \begin{pmatrix} Aw - (\gamma-1)e_m \\ -Bw + (\gamma+1)e_k \end{pmatrix}_+ = 0 \tag{23}$$

The last scalar equation from above gives

$$e_m^T(Aw - (\gamma - 1)e_m)_+ = e_k^T(-Bw + (\gamma + 1)e_k)_+ \tag{24}$$

or equivalently

$$\sum_{i=1}^{m} \left[a_i^T w - (\gamma - 1)\right]_+ = \sum_{j=1}^{k} \left[-b_j^T w + (\gamma + 1)\right]_+ , \tag{25}$$

which is the statement of the proposition. □ The one-dimensional minimization is easily carried out by searching through the knot points defined by $\gamma_i = a_i^T w$ and $\gamma_j = b_j^T w$ and adds little to the overall computation costs. This usually improves the solution by a few data points for two-dimensional test problems, but is much less effective for higher dimensional problems. Intuitively, in the two-dimensional case minimizing on γ varies one third of the variables, while for a 9 or 13 dimensional problem, only one tenth or fourteenth of the variables are being changed.

7. Performance on Test Databases. The linear separability methods have been tested on the Wisconsin Breast Cancer and Cleveland Heart Disease Databases [3]. Here the goal is to provide a linear predictor that can be used to distinguish between benign and malignant tumors in the first data base, and patients at risk or not at risk of heart attack in the second data base.

Both data sets are available from the University of California–Irvine Repository Of Machine Learning Databases and Domain Theories[2]. The first data set consists of 551 points, 346 from set $\mathcal{A}$ (corresponding to benign tumors) and 205 from set $\mathcal{B}$ (corresponding to malignant tumors). Each data point has 9 components, corresponding to experimental measurements. The second data set consists of 297 points, 137 from set $\mathcal{A}$ (corresponding to a negative diagnosis) and 160 from set $\mathcal{B}$ (corresponding to a positive diagnosis).

We discarded data samples that had missing measurements. As in [1], the data was divided randomly into a training group consisting of 2/3 of the data points, and a testing group consisting of the remaining 1/3 of the data. The best separating hyperplane was found by applying the inequality least squares solver using the training group of data, and then the effectiveness of the hyperplane was tested on the remaining testing group of data. This was repeated ten times, with different partitionings into training and testing sets. For the first data set, the inequality solver required 7 or 8 iterations each time and spent an average of 92% of its flops in finding the search direction. For the second data set, 5 or 6 iterations were needed each time and an average of 95% of the flops were spent in finding the search direction.

[2] via anonymous ftp to ics.uci.edu, in directory pub/machine-learning-databases

Database	Data Group	Ls Sq w/o γ Min	Ls Sq w γ Min	Lin Progr
Cancer	Training	3.19	2.64	3.01
	Testing	4.24	3.80	2.56
Heart	Training	14.75	13.84	15.23
	Testing	15.76	15.96	16.53

TABLE 3
Percent of Incorrectly Classified Points For Two Databases

Table 3 shows the results both with and without the secondary minimization on γ, along with similar results from [1]. The results are not strictly comparable since that work used 566 data points from the Wisconsin Cancer Database and 197 data points from the Cleveland Heart Disease Database, but the comparison suggests that the inequality least squares method provides a solution similar to that of a linear programming method. This is unexpected because the two–norm solution given by inequality least squares can heavily weight outlying data points while the one–norm solution given by the linear programming method weights outlying data points less. However, these results show that for realistic problems the inequality least squares solution is qualitatively competitive.

8. Summary and Future Work. A computationally efficient implementation of Han's algorithm for solving linear inequalities in a least squares sense has been presented, and has been shown convergent by minor modifications to his convergence proofs. The effectiveness of this change in the algorithm's implementation has been demonstrated, and it has been tested on randomly generated and linear separability problems. The results indicate that with the new implementation, this linear inequalities solution method is a worthwhile addition to a computational scientist's toolkit.

Current work includes applying this method to the graph partitioning problem for parallel computing (see [7] for a survey of this problem and solution methods). This works in phases: given a graph corresponding to a physical mesh, first find the "deepest" set of nodes by a breadth first search from the boundary nodes. Two nodes that are furthest apart in the deepest set are selected as initial $\mathcal{A}$ and $\mathcal{B}$ points, and the sets are grown outwards using a constrained breadth first search. A separating hyperplane is then found to better define those sets, followed possibly by a few steps of simulated annealing to further improve the partitioning. This avoids finding eigenvalues or singular values of the Laplacian of a sparse matrix, and uses only fast graph algorithms and a relatively low cost linear inequality solve.

The most interesting work remaining is a proof of convergence within $\max(m, n) + 1$ iterations, or a counterexample. Since QR factorization with column pivoting takes

$\mathcal{O}(mn^2)$ work in the dense case, this would provide a polynomial upper bound on the algorithm. A linear programming problem can be stated as a system of inequalities [6], so this algorithm would be another polynomial method for linear programming. Furthermore, the subproblems generated by the inequality least squares algorithm have condition numbers no worse than that of the original data, since at each step a decomposition is performed on a submatrix of A (condition number for least squares problems is defined as the ratio of largest to smallest nonzero singular values). Interior point methods for linear programming have subproblems that become increasingly ill-conditioned as the iterates converge. So if the conjectured upper bound on the number of iterations holds, this algorithm provides a numerically stable polynomial time algorithm for linear programming.

REFERENCES

[1] K. Bennett and O. Mangasarian, *Robust linear programming discrimination of two linearly inseparable sets*, Optimization Methods and Software, 1 (1992), pp. 23–34.

[2] R. Bramley and B. Winnicka, *Solving linear inequalities in a least squares sense*, Tech. Rep. 396, Indiana University–Bloomington, Bloomington, IN 47405, 1995.

[3] R. Detrano et al., *International application of a new probability algorithm for the diagnosis of coronary artery disease*, American Journal of Cardiology, 64 (1989), pp. 304–310.

[4] G. Golub and C. V. Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, 2 ed., 1989.

[5] G. Golub and V. Pereysa, *Differentiation of pseudoinverse, separable nonlinear least squares problems and other tales*, in Generalized Inverses and Applications, M. Nashed, ed., Academic Press, New York, 1976, pp. 303–323.

[6] S.-P. Han, *Least–squares solution of linear inequalities*, Tech. Rep. TR–2141, Mathematics Research Center, University of Wisconsin-Madison, 1980.

[7] B. Hendrickson and R. Leland, *The Chaco user's guide*, tech. rep., Sandia National Laboratory, Albuquerque, N.M., 1994. email rwlelan@cs.sandia.gov.

[8] S. Katznelson, *An algorithm for solving nonlinear resistor networks*, Bell System Technical Journal, 44 (1965), pp. 1605–1620.

[9] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice–Hall, Englewood Cliffs, NJ, 1974.

[10] W. Li and J. Swetits, *A Newton method for solving convex quadratic programs*, SIAM Journal on Optimization, 3 (1993), pp. 466–488.

[11] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[12] R. Penrose, *A generalized inverse for matrices*, Proc. Cambridge Phil. Soc., 51 (1955), pp. 406–413.

[13] G. Stewart, *An iterative method for solving linear inequalities*, Tech. Rep. TR–1833, University of Maryland Computer Science Department, 1987.

[14] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.