Parsimonious Least Norm Approximation [*]

P. S. Bradley[†] O. L. Mangasarian[‡] & J. B. Rosen[§]

Abstract

A theoretically justifiable fast finite successive linear approximation algorithm is proposed for obtaining a parsimonious solution to a corrupted linear system $Ax = b + p$, where the corruption p is due to noise or error in measurement. The proposed linear-programming-based algorithm finds a solution x by parametrically minimizing the number of nonzero elements in x and the error $\| Ax - b - p \|_1$. Numerical tests on a signal-processing-based example indicate that the proposed method is comparable to a method that parametrically minimizes the 1-norm of the solution x and the error $\| Ax - b - p \|_1$, and that both methods are superior, by orders of magnitude, to solutions obtained by least squares as well by combinatorially choosing an optimal solution with a specific number of nonzero elements.

Keywords Minimal cardinality, least norm approximation

1 Introduction

A wide range of important applications can be reduced to the problem of estimating a vector x by minimizing some norm of the residual vector $Ax - b$ arising from a possibly inconsistent system of linear equations:

$$Ax = b, \tag{1}$$

where A is an $m \times n$ real matrix and b is an $m \times 1$ vector, and both A and b are subject to error. Methods for solving such problems include least squares [15], total least squares [11, 14] and structured total least norm [24, 13].

In this paper we consider the closely related problem of minimizing the 1-norm of the residual vector $Ax - b$, where b is subject to error and with the additional condition that only a specified number $k < n$ of columns of A are used. This is clearly a combinatorial problem which is closely related to the NP-hard problem considered by Amaldi and Kann [2] and consisting of solving a consistent system of linear inequalities or equalities with rational entries such that the solution x has a minimal number of nonzeros. We shall solve our problem by a novel method, based on minimizing a concave function on a polyhedral set, that has been successfully used in such machine learning problems as misclassification minimization [17], and feature selection [6] and in data mining [5, 19].

[*]Mathematical Programming Technical Report 97-03, March 1997 – Revised September & November 1997. This material is based on research supported by National Science Foundation Grants CCR-9322479 and CCR-9509085, and Air Force Office of Scientific Research Grant AFOSR-144-GC92.

[†]Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706, *paulb@cs.wisc.edu*

[‡]Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706, *olvi@cs.wisc.edu.* This author gratefully acknowledges the gracious hospitality of the Mathematics Department of the University of California at San Diego during his sabbatical leave January-May 1997.

[§]Computer Science & Engineering, University of California San Diego, La Jolla, CA 92093 *jbrosen@ucsd.edu*

The idea behind using as few columns of A as possible to span b is motivated by the parsimony principle of machine learning, known as Occam's Razor [26, 4], which says in essence: "simplest is best". This principle is highly effective for generalization purposes [16, 25, 30] where, for example, one wishes to use the "solution" x of (1) on new data not represented by the rows of $[A\ b]$ as would be the case if either A or b is corrupted by noise. The use of the 1-norm will enable us to use a finite algorithm based on the polyhedral concave minimization approach which, as indicated above, has been successfully used on difficult machine learning problems. In particular we will eventually cast the problem as that of minimizing a concave function on a polyhedral set and begin with the following unconstrained minimization problem:

$$\min_{x \in R^n} (1-\lambda)\|Ax-b\|_1 + \lambda e'|x|_*, \quad \lambda \in [0,1) \tag{2}$$

Here e is a column vector of ones, the prime denotes the transpose, $|\cdot|$ denotes the absolute value function applied componentwise to a vector and $(\cdot)_*$ is the step function applied componentwise also. The step function takes the value 0 if its argument is nonpositive, and the value 1 if its argument is positive. The vector b will be corrupted by a noise vector p in our application. We note immediately that when $\lambda = 0$, problem (2) is the classical least 1-norm approximation problem. When $\lambda = 1$, problem (2) is trivially solved by $x = 0$ and is of no interest. We are interested in solutions to problem (2) with $\lambda \in [0,1)$ that make $e'|x|_* \leq k$ for some desired $k < n$ and such that $\|Ax-b\|_1$ is acceptably small. In fact problem (2) can be viewed as a multiobjective optimization problem [8] with the the two objectives of parsimony in the number of nonzero components of x and smallness of the error $\|Ax-b\|_1$. By letting λ range over the interval $[0,1]$ the cardinality of the nonzero elements of the solution x varies from a maximum of n to 0, while the error $\|Ax-b\|_1$ will be nondecreasing monotonically. Depending on the problem, one of those x's will be the most desirable. In many of the machine learning applications small values of λ such as 0.05 often gave parsimonious results that improved tenfold cross-validation [6]. We shall call problem (2), with a possibly noise-corrupted b, the parsimonious least norm approximation problem (PLNA).

Our approach here for solving (2) will be to convert it to a concave minimization problem on a polyhedral set (problem (12) below). We first show that this problem always has a solution (Theorem 2.1 below). We then replace the discontinuous step function in the objective function of (12) below by an exponential smooth function in problem (14) below, just as was done in [18, 6], and relate the two problems. Our novel theorem (Theorem 2.1 below) shows that the continuous problem yields an exact solution of the discontinuous problem once a repeating optimal vertex is identified for increasing but finite values of the smoothing parameter α. We then prescribe a linear-programming-based successive linearization algorithm SLA 3.1 for the solution of the smooth problem and establish its finite termination in Theorem 3.2.

For comparative purposes we shall also employ Vapnik's support vector machine approach [29, 3] of minimizing the size of the solution vector x as well as the error $\|Ax-b\|_1$, thereby decreasing the VC dimension [29, p 76] (a capacity measure) and improving generalization. We shall do that by parametrically minimizing the 1-norm of x as well as the 1-norm of the error $Ax-b$:

$$\min_{x \in R^n} (1-\lambda)\|Ax-b\|_1 + \lambda\|x\|_1 \tag{3}$$

We shall call this problem, with a possibly noise-corrupted b, the *least* least norm approximation (LLNA) problem and solve it by solving the equivalent linear programming formulation:

$$\min_{(x,y,z) \in R^{n+m+n}} \{(1-\lambda)e'y + \lambda e'z \mid -y \leq Ax-b \leq y,\ -z \leq x \leq z\} \tag{4}$$

A word about our notation and background material. All vectors will be column vectors unless transposed to a row vector by a prime superscript $'$. For a vector x in the n-dimensional real space R^n, $|x|$ will denote a vector of absolute values of components x_i, $i = 1, \ldots, n$ of x. The scalar product of two vectors x and y in the n-dimensional real space will be denoted by $x'y$. For a linear program $\min\limits_{x \in X} c'x$, the notation arg vertex $\min\limits_{x \in X} c'x$ will denote the set of vertex solutions of the linear program. For $x \in R^n$, the norm $\|x\|_2$ will denote the 2-norm: $(x'x)^{\frac{1}{2}}$, while $\|x\|_1$ will denote the 1-norm: $\sum\limits_{i=1}^{n} |x_i|$. For an $m \times n$ matrix A, A_i will denote the ith row of A and A_{ij} will denote the element in row i and column j. The identity matrix in a real space of arbitrary dimension will be denoted by I, while a column vector of ones of arbitrary dimension will be denoted by e. The base of the natural logarithm will be denoted by ε, and for $y \in R^m$, ε^{-y} will denote a vector in R^m with component ε^{-y_i}, $i = 1, \ldots, m$. For a function $f : R^n \to R$ that is concave on R^n, the supergradient $\partial f(x)$ of f at x is a vector in R^n satisfying

$$f(y) - f(x) \leq (\partial f(x))'(y - x) \tag{5}$$

for any $y \in R^n$. The set $D(f(x))$ of supergradients of f at the point x is nonempty, convex, compact and reduces to the ordinary gradient $\nabla f(x)$, when f is differentiable at x [22, 23]. For a vector $x \in R^n$, card(x) will denote the cardinality of the nonzero elements of x.

2 The Concave Minimization Problem

In this section we shall consider the minimization problem

$$\min_{s \in S} f(s) + \mu h'|s|_*, \tag{6}$$

where f is a concave function on R^k which is bounded below on S, μ is a nonnegative real number, h is a nonnegative vector in R^k and S is a polyhedral set in R^k not containing straight lines that go to infinity in both directions. Note that if the objective function of (6) is concave (which it is not in general because of the nonconcavity of $h'|s|_*$) then by [23, Corollary 32.3.3] problem (6) has a solution and by [23, Corollary 32.3.4] it has a vertex solution since S contains no straight lines that go to infinity in both directions. However despite this lack of concavity we shall show precisely the existence of a vertex solution by a novel approach which approximates the step function on the nonnegative real line from below by an exponential. This smooth approximation will also serve as a means for generating a finitely terminating algorithm at a stationary point of (6). Another important feature is that an exact solution of (6) is obtained from a solution of the smooth approximation for a finite value of the smoothing parameter.

We state now our smooth approximation of (6) as follows

$$\min_{s \in S} f(s) + \mu h'(e - \varepsilon^{-\alpha|s|}), \tag{7}$$

where α is a positive number. We have the obvious relation

$$h'|s|_* \geq h'(e - \varepsilon^{-\alpha|s|}), \ \forall s \in R^k. \tag{8}$$

Hence the smooth problem (7) minimum provides an underestimate to the minimum of problem (6). This fact will be used to establish exact solution of the latter by the former in the following principal theorem of the paper which also provides a method of solution as well.

Theorem 2.1 Existence of Exact Vertex Solution for Finite Value of Smoothing Parameter *Let $f : R^k \longrightarrow R$ be bounded below on the polyhedral set S that contains no straight lines going to infinity in both directions, let f be concave on R^k, let $h \geq 0$ and let μ be a fixed positive number. Then for a sufficiently large positive but finite value α_0 of α, the smooth problem (7) has a vertex solution that also solves the original nonsmooth problem (6).*

Proof Note first that the smooth problem (7) is equivalent to the following concave minimization problem

$$\min_{(s,z)\in T} f(s) + \mu h'(e - \varepsilon^{-\alpha z}), \text{ where } (s,z) \in T := \{(s,z) \mid s \in S,\ -z \leq s \leq z\}. \tag{9}$$

Since the objective function of this problem is concave in (s,z) on R^{2k} and is bounded below on T, it follows by [23, Corollaries 32.3.3 and 32.3.4] that it has a vertex $(s(\alpha), z(\alpha))$ of T as a solution for each $\alpha > 0$. Since T has a finite number of vertices, one vertex, say $(\bar{s}, \bar{z})$, will repeatedly solve problem (9) for some sequence $\{\alpha_0, \alpha_1, \ldots\} \uparrow \infty$. Hence for $\alpha_i \geq \alpha_0$,

$$\begin{array}{rcl} f(\bar{s}) + \mu h'(e - \varepsilon^{-\alpha_i \bar{z}}) & = & f(s(\alpha_i)) + \mu h'(e - \varepsilon^{-\alpha_i z(\alpha_i)}) \\ & = & \displaystyle\min_{(s,z)\in T} f(s) + \mu h'(e - \varepsilon^{-\alpha_i z}) \\ & = & \displaystyle\min_{s\in S} f(s) + \mu h'(e - \varepsilon^{-\alpha_i |s|}) \\ & \leq & \displaystyle\inf_{s\in S} f(s) + \mu h'|s|_*, \end{array} \tag{10}$$

where the last inequality follows from (8). Letting $i \longrightarrow \infty$ it follows that

$$f(\bar{s}) + \mu h'|\bar{s}|_* = \lim_{i\longrightarrow\infty} f(\bar{s}) + \mu h'(e - \varepsilon^{-\alpha_i \bar{z}}) \leq \inf_{s\in S} f(s) + \mu h'|s|_*. \tag{11}$$

Since $\bar{s} \in S$, it follows that $\bar{s}$ solves (6). Since $(\bar{s}, \bar{z})$ is a vertex of T, it follows that $\bar{s}$ is a vertex of S. $\square$

This theorem immediately suggests an algorithmic approach for solving our problem (2) as follows. We first rewrite (2) as the following equivalent problem

$$\min_{(x,y)\in S} (1-\lambda)e'y + \lambda e'|x|_*, \quad S := \{(x,y) \mid x \in R^n,\ y \in R^m,\ -y \leq Ax - b \leq y\},\ \lambda \in [0,1) \tag{12}$$

By making the identifications

$$s = \begin{pmatrix} x \\ y \end{pmatrix},\ k = n + m,\ f(s) = e'y,\ \mu = \frac{\lambda}{1-\lambda},\ h = \begin{pmatrix} e \\ 0 \end{pmatrix}, \tag{13}$$

problem (12) and hence problem (2) becomes a special case of problem (6) which we shall solve in its smooth version (7). More specifically the smooth version of (2) is the following concave minimization problem:

$$\min_{(x,y,z)\in T} (1-\lambda)e'y + \lambda e'(e - \varepsilon^{-\alpha z}), \text{ where } T := \{(x,y,z) \mid -y \leq Ax - b \leq y,\ -z \leq x \leq z\}. \tag{14}$$

By solving this problem for a sufficiently large positive α it follows by Theorem 2.1 that we have solved our original problem (2). We turn our attention now to solving (14) by a finitely terminating successive linearization algorithm.

3 The Concave Minimization Algorithm

The finite method that we shall propose is the successive linear approximation (SLA) method of minimizing a concave function on a polyhedral set which is a finitely terminating stepless Frank-Wolfe algorithm [9]. In [18] finite termination of the SLA was established for a differentiable concave function, and in [20] for a nondifferentiable concave function using its supergradient. We state now the SLA for problem (14) which has a differentiable concave objective function.

Algorithm 3.1 Successive Linearization Algorithm (SLA) *Start with a random* $x^0 \in R^n$, $y^0 = |Ax^0 - b|$, $z^0 = |x^0|$. *Having* (x^i, y^i, z^i) *determine*

$$(x^{i+1}, y^{i+1}, z^{i+1}) \in \arg \min_{(x,y,z)\in T} (1-\lambda)e'y + \lambda\alpha(\varepsilon^{-\alpha z^i})'z \tag{15}$$

Stop when $(x^i, y^i, z^i) \in T$ *and*

$$(1-\lambda)e'y^i + \lambda\alpha(\varepsilon^{-\alpha z^i})'z^i = (1-\lambda)e'y^{i+1} + \lambda\alpha(\varepsilon^{-\alpha z^i})'z^{i+1} \tag{16}$$

By [18, Theorem 4.2] we have the following finite termination result for the SLA algorithm.

Theorem 3.2 SLA Finite Termination *The SLA 3.1 generates a finite sequence* $\{(x^i, y^i, z^i)\}$ *with strictly decreasing objective function values for problem (14) and terminating at an* $\bar{i}$ *satisfying the minimum principle necessary optimality condition*

$$(1-\lambda)e'(y - y^{\bar{i}}) + \lambda\alpha(\varepsilon^{-\alpha z^{\bar{i}}})'(z - z^{\bar{i}}) \geq 0, \ \forall\, (x,y,z) \in T \tag{17}$$

4 Application and Numerical Testing

We wish to determine whether x-component suppression or x-norm reduction of an observed linear system $Ax = b+p$ which is a corruption of a true system $Ax = b$, leads to an improved approximation of the true system. One can relate this to a machine learning framework by treating the first system as a *training set*, and the second system as a *testing set* [12]. The linear systems used are based upon ideas related to signal processing [10, 28] and more specifically to an example in [1, Equation (8)].

We consider the following *true* signal $g(t) : [0,1] \longrightarrow R$:

$$g(t) = \sum_{j=1}^{3} x_j \varepsilon^{-a_j t}, \ t \in [0,1], \ a = [0 \ \ 4 \ \ 7]', \ x = [0.5 \ \ 2.0 \ \ -1.5]'. \tag{18}$$

We assume that the true signal $g(t)$ cannot be sampled precisely, but that the following *observed* signal can be sampled:

$$\tilde{g}(t) = (g(t) + \text{error}), \ \text{ sampled at times}: \ t_i = i \bigtriangleup t, \ \bigtriangleup t = 0.04, \ i = 0, 1, \ldots, 25. \tag{19}$$

We further assume that we do not know the true signal $g(t)$ (18), and we attempt to model it as:

$$\hat{g}(t) = \sum_{j=1}^{10} x_j \varepsilon^{-a_j t}, \ t \in [0,1], \ a = [0 \ \ 4 \ \ 7 \ \ 0.1 \ \ 2 \ \ 3 \ \ 3.9 \ \ 4.1 \ \ 6.9 \ \ 7.1]'. \tag{20}$$

The problem now is to compute the coefficients x_j, $j = 1, \ldots, 10$, of $\hat{g}(t)$ (20) so that we can adequately recover $g(t)$, given only the noisy data $\tilde{g}(t_i)$ (19). Notice that by substituting the following coefficient vector x^* into (20), $\hat{g}(t) = g(t)$:

$$x^* := [0.5 \;\; 2.0 \;\; -1.5 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]'. \tag{21}$$

Thus the *true* linear system (testing set) $Ax = b$ is then given by:

$$A_{ij} = \varepsilon^{-a_j t_i},\; b_i = g(t_i),\; i = 0, \ldots, 25,\; j = 1, \ldots 10, \tag{22}$$

and is solved exactly by x^* of (21).

The *observed* linear system (training set) $Ax = b + p$ is then given by:

$$\left\langle \begin{array}{l} A_{ij} = \varepsilon^{-a_j t_i}, \;\; b_i = g(t_i), \\ p_i = \text{random number with mean } = 0 \text{ \& standard deviation } = 1, \\ i = 0, \ldots, 25,\; j = 1, \ldots, 10. \end{array} \right\rangle \tag{23}$$

We will refer to a solution of problem (14), with b of (14) replaced by $b + p$, computed by the Successive Linearization Algorithm (SLA 3.1) as a PLNA solution. Similarly, we shall refer to a solution of problem (4), with b replaced by $b + p$ as an LLNA solution. We note here that for *all* experiments, the value of α in the negative exponential of (14) is 5.0. Scalars are considered zero if they are in the interval $[-1e - 8, 1e - 8]$. The components of the initial starting point x^0 for SLA 3.1 were sampled from a normal distribution with mean = 0 and standard deviation = 1. The components of the initial point were sampled then *fixed* for all runs as:

$$x^0 = [-0.5077 \;\; 0.8853 \;\; -0.2481 \;\; -0.7262 \;\; -0.4450 \;\; -0.6129 \;\; -0.2091 \;\; 0.5621 \;\; -1.0639 \;\; 0.3516]'. \tag{24}$$

We now focus our attention on four approaches and compare solutions obtained by the PLNA and LLNA methods with solutions obtained by least squares and by a combinatorial search.

4.1 Comparison of PLNA, LLNA and Least Squares

We compute solutions of the observed system $Ax = b + p$, where A, b, and p are defined in (23), by PLNA, LLNA and by least squares. These solutions are then evaluated by the observed system (training set) residual $\|Ax - b - p\|_1$ and the true system (testing set) residual $\|Ax - b\|_1$ and graphically comparing the recovered signal $\hat{g}(t)$ (20) to the true signal $g(t)$ (18).

The PLNA solution $x(\lambda)$ of $Ax = b + p$, for a given λ is computed by solving by SLA 3.1 the concave minimization problem (14) with b replaced by $b + p$ as follows:

$$\min_{(x,y,z) \in T} (1 - \lambda) e'y + \lambda e'(e - \varepsilon^{-\alpha z}),\; T := \{(x, y, z) \mid -y \le Ax - b - p \le y,\; -z \le x \le z\}. \tag{25}$$

The LLNA solution $x(\lambda)$ of $Ax = b + p$, for a given λ is computed by solving the linear program (4) with b replaced by $b + p$ as follows:

$$\min_{(x,y,z) \in R^{n+m+n}} \{(1 - \lambda) e'y + \lambda e'z \mid -y \le Ax - b - p \le y,\; -z \le x \le z\}. \tag{26}$$

The least squares solution is a minimizer of $||Ax - b - p||_2$ and is a solution to the normal equations:

$$A'Ax = A'(b+p). \tag{27}$$

Although the 26×10 matrix A defined by (23) has rank 10, the matrix $A'A$ is numerically singular with smallest eigenvalue less than 10^{-14}. Thus we resort to a singular value decomposition approach for solving (27).

We determine an approximate solution $x(ls)$ to (27) by the following method which utilizes the singular value decomposition [27]. Ordinary MATLAB [21] commands such as $x = A\backslash(b+p)$ for our perturbed system $Ax = b + p$ give an x with an error $||x - x^*||_2 = 2.1379e + 08$ compared to $||x - x^*||_2 = 2.6675e + 03$ given by the method described below, where x^* is efined by (21) and the perturbation vector p components are sampled from a normal distribution with mean = 0, standard deviation = 1.

Algorithm 4.1 Least Squares via Singular Value Decomposition. *Let $A \in R^{m\times n}$ with $m \geq n$. Let τ be a small positive tolerance.*

1. *Determine the economy singular value decomposition of A [21, svd(A,0)], $U \in R^{m\times n}$, $S \in R^{n\times n}, V \in R^{n\times n}$:*

$$A = USV', \tag{28}$$

where $U'U = V'V = I_n$ (the $n \times n$ identity matrix), and $S = diag(\sigma_1, \ldots, \sigma_n)$, $\sigma_i \geq \sigma_{i+1} \geq 0$, $i = 1, \ldots, n-1$.

2. *Determine the index r such that $\sigma_i \geq \tau$ for $i = 1, \ldots, r$.*

3. *Set $\tilde{U} \in R^{m\times r}$ to be the first r columns of U, $\tilde{V} \in R^{n\times r}$ to be the first r columns of V and $\tilde{S} \in R^{r\times r}$ to be $diag(\sigma_1, \ldots, \sigma_r)$.*

4. *Compute $x(ls) = \tilde{V}\tilde{S}^{-1}\tilde{U}'(b+p)$, which is a solution to:*

$$\min_{x\in\tilde{T}} \frac{1}{2}x'x, \quad \tilde{T} := \{x \mid \tilde{V}'x = \tilde{S}^{-1}\tilde{U}'(b+p)\ \} \approx \{x \mid A'Ax = A'(b+p)\}. \tag{29}$$

For all runs τ was fixed at 0.0001, which for our specific matrix A defined by (23), led to $r = 6$ in the above algorithm. That is we discarded the last 4 columns of U and V.

The PLNA problem (25) and the LLNA problem (26) were both solved for values of $\lambda \in \{0, 0.01, 0.05, 0.10, 0.20, \ldots, 0.90, 0.95, 0.99, 1.0\}$. Figures 1 - 3 display results averaged over 5 noise vectors $p \in R^m$ with elements sampled from a normal distribution with mean = 0, standard deviation = 1. The average $||p||_1 = 21.1008$ and $||b||_1 = 20.1777$.

In Figure 1 we plot averages of $||Ax(\lambda) - b - p||_1$ for the various values of λ, measuring how "well" the PLNA and LLNA solutions solve the corrupted observed linear system. Also plotted is the average of $||Ax(ls) - b - p||_1$, measuring how "well" the least squares solution (Algorithm 4.1) solves the observed system $Ax = b + p$. As can be proved, the PLNA and LLNA errors are a non–decreasing functions of λ and are worse than the corresponding least squares error. However on the true system the results are reversed. See next paragraph.

In Figure 2 we plot averages of $||Ax(\lambda) - b||_1$ for both PLNA and LLNA for various values of λ, measuring how "well" the PLNA solution (25) solves the true linear system. Also plotted is the

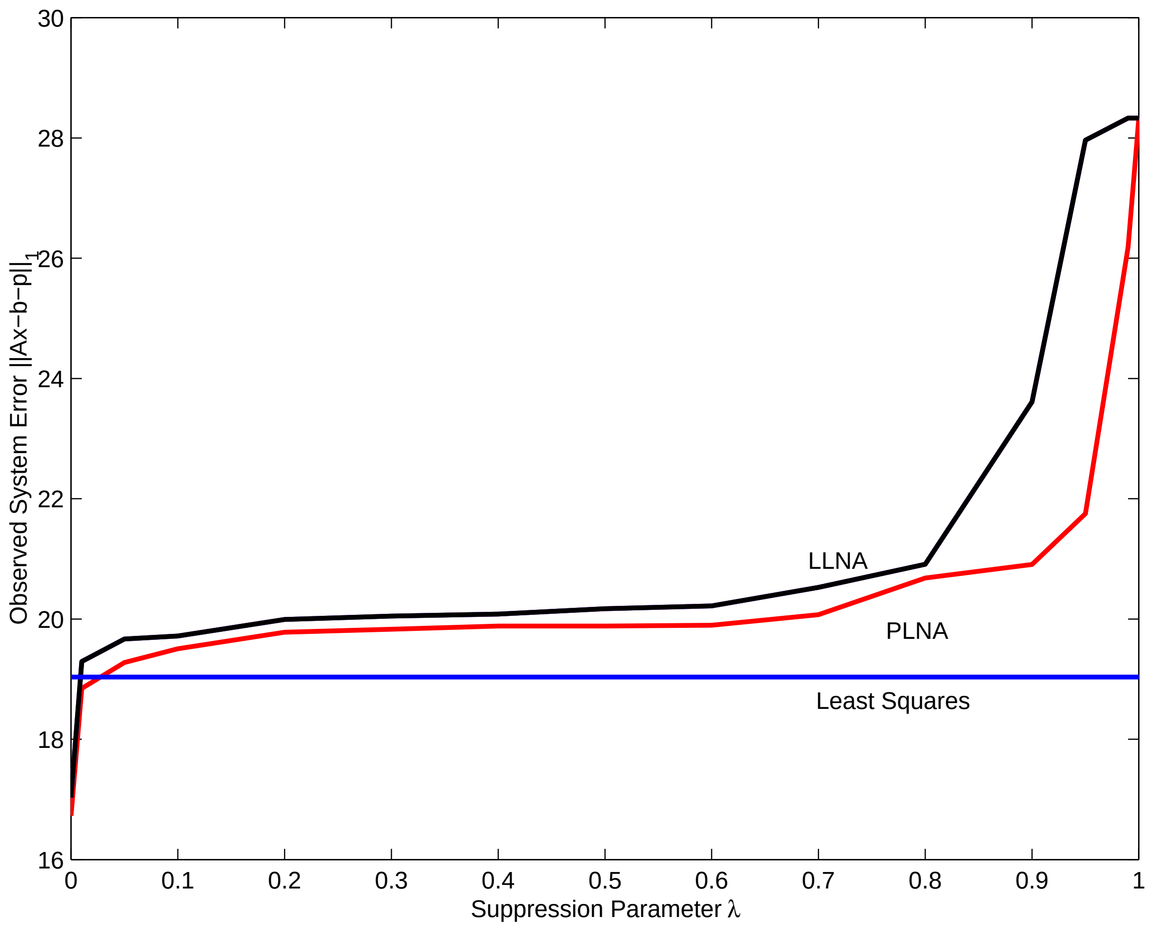


Figure 1: Average $\|Ax(\lambda) - b - p\|_1$ versus λ, where $x(\lambda)$ is a PLNA solution (25) in the curve marked PLNA and is an LLNA solution of (26) for the curve marked LLNA, compared with average $\|Ax(ls) - b - p\|_1$, where $x(ls)$ is the least squares solution (27) by Algorithm 4.1. The results are averaged over 5 noise vectors p. The PLNA and LLNA solutions were computed for values of $\lambda = 0, 0.01, 0.05, 0.10, 0.20, \ldots, 0.90, 0.95, 0.99, 1$.

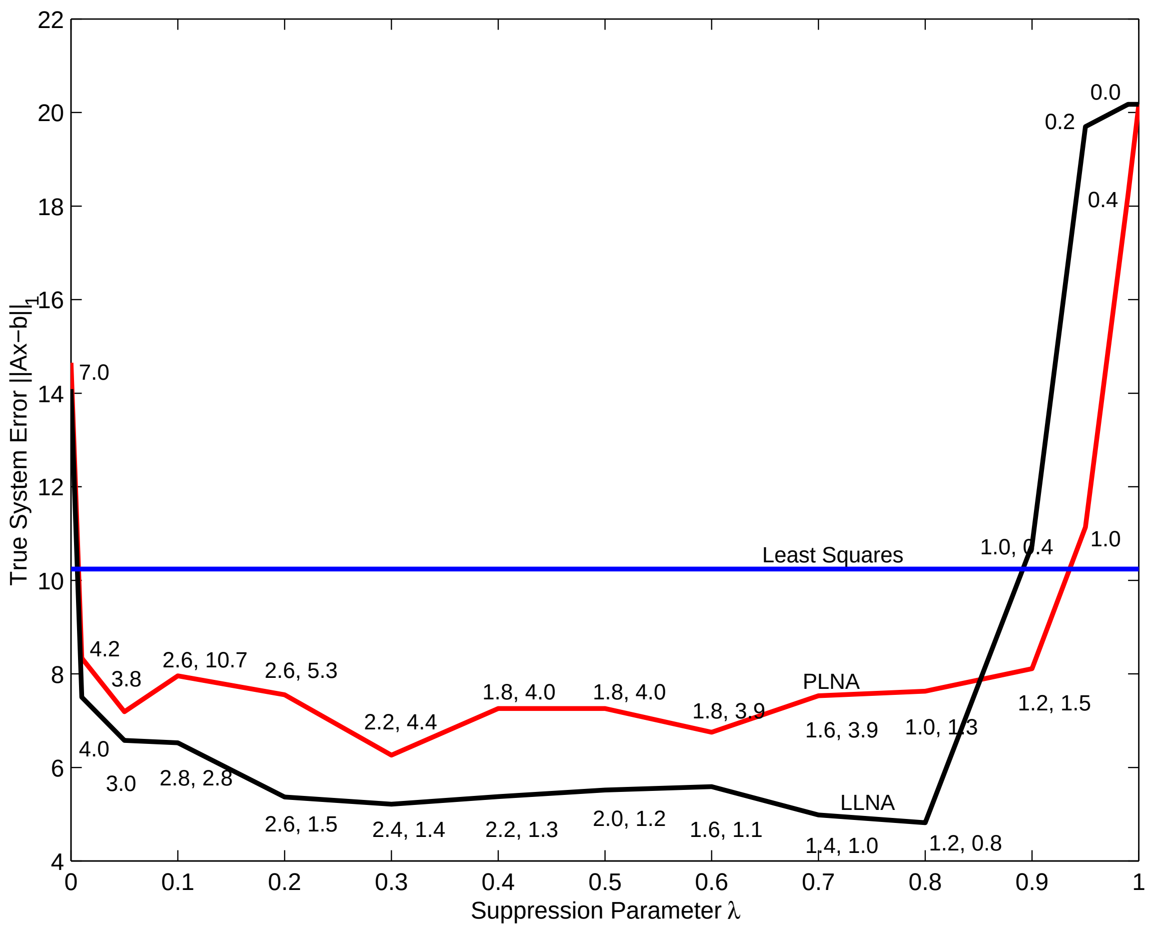


Figure 2: Average $\|Ax(\lambda) - b\|_1$ versus λ, where $x(\lambda)$ is a PLNA solution (25) in the curve marked PLNA and is an LLNA solution of (26) for the curve marked LLNA, compared with the average $\|Ax(ls) - b\|_1$, where $x(ls)$ is the least squares solution (27) solved by Algorithm 4.1. These results are averaged over 5 noise vectors p. The PLNA and LLNA solutions were computed for values of $\lambda = 0, 0.01, 0.05, 0.10, 0.20, \ldots, 0.90, 0.95, 0.99, 1$. Numbers above/below the curves labelled "PLNA" and "LLNA" at various values of λ indicate the average number of nonzero elements in $x(\lambda)$ and when followed by a second number, that number denotes $\|x(\lambda)\|_1$.

average of $\|Ax(ls) - b\|_1$, measuring how "well" the least squares solution (Algorithm 4.1) solves $Ax = b$.

In Figure 3 we compare averages of 1-norm distances from the true solution x^* (21) to the PLNA and LLNA solutions $x(\lambda)$ and the averages of 1-norm distances from x^* to the least squares solution $x(ls)$. Recall that the true solution x^* is such that $Ax^* = b$. Note that for $\lambda \geq 0.01$, the PLNA and LLNA distances are smaller than the least squares distance. For $\lambda \approx 1$, $x(\lambda) \approx 0$ and even though $\|x(\lambda) - x^*\|_1$ is small, this solution is poor from a signal recovery point of view since the zero vector gives the worst discrepancy between the true signal and the recovered signal at 26 discrete points (see Figure 2).

In Figure 4(a) we plot the true signal, the observed signal and the signal recovered by solving, for one noise vector p, PLNA (25) with $\lambda = 0.30$ and LLNA (26) for $\lambda = 0.80$. Figure 4(b) displays the true signal, the observed signal and signal recovered for the same problem by least squares (27) solved by Algorithm 4.1. This is probably the most significant result. The signal recovered by both PLNA and LLNA is considerably closer to the the true signal than that obtained by the least squares solution.

4.2 Comparison of PLNA and LLNA with Combinatorial Search

In this section, we reformulate our PLNA problem so that the solution $x(\lambda)$ has a fixed number of nonzero elements, for $k \in \{1, 2, \ldots, n\}$:

$$(x(\lambda), y(\lambda), z(\lambda)) \in \arg \min_{(x,y,z)\in T} (1-\lambda)e'y + \lambda e'\varepsilon^{-\alpha z} \tag{30}$$

$$T := \left\{ (x, y, z) \;\middle|\; \begin{array}{l} -y \leq Ax - b - p \leq y, \; -z \leq x \leq z, \\ \# \text{ of nonzero elements of } x = k \end{array} \right\}$$

We also formulate the LLNA similarly as follows:

$$(x(\lambda), y(\lambda), z(\lambda)) \in \arg \min_{(x,y,z)\in T} (1-\lambda)e'y + \lambda e'z \tag{31}$$

Similarly, for $k \in \{1, 2, \ldots, n\}$, the combinatorial search solution x_c is obtained by solving:

$$x_c \in \arg \min_{x\in R^n} \{\|Ax - b - p\|_1 \mid \# \text{ of nonzero elements of } x = k\}. \tag{32}$$

Notice that x_c is determined by enumerating all subsets of size k of a set of n elements, or $\binom{n}{k}$ subsets. This is a rather expensive procedure computationally requiring two orders of magnitude more time than PLNA and LLNA.

Figure 5 displays results averaged over 5 noise vectors $p \in R^m$ with elements sampled from a normal distribution with mean = 0, standard deviation = 1 (average $\|p\|_1 = 21.1008$, $\|b\|_1 = 20.1777$). Plotted are averages of $\|Ax(\lambda) - b - p\|_1$ and $\|Ax_c - b - p\|_1$ for each k measuring how "well" the PLNA, LLNA and combinatorial solutions solve the observed system. Also plotted are averages of $\|Ax(\lambda) - b\|_1$ and $\|Ax_c - b\|_1$ for each k, measuring how "well" the solutions solve the true system.

Figure 6 displays the average 1-norm distance between x^* of (21) and the solutions obtained by PLNA, LLNA and combinatorial search. The averages are over 5 noise vectors p.

Figure 7(a), which for convenience duplicates Figure 4(a), displays the true signal, the observed signal and the signal recovered by solving PLNA (25) for the value of $\lambda = 0.30$ and the signal

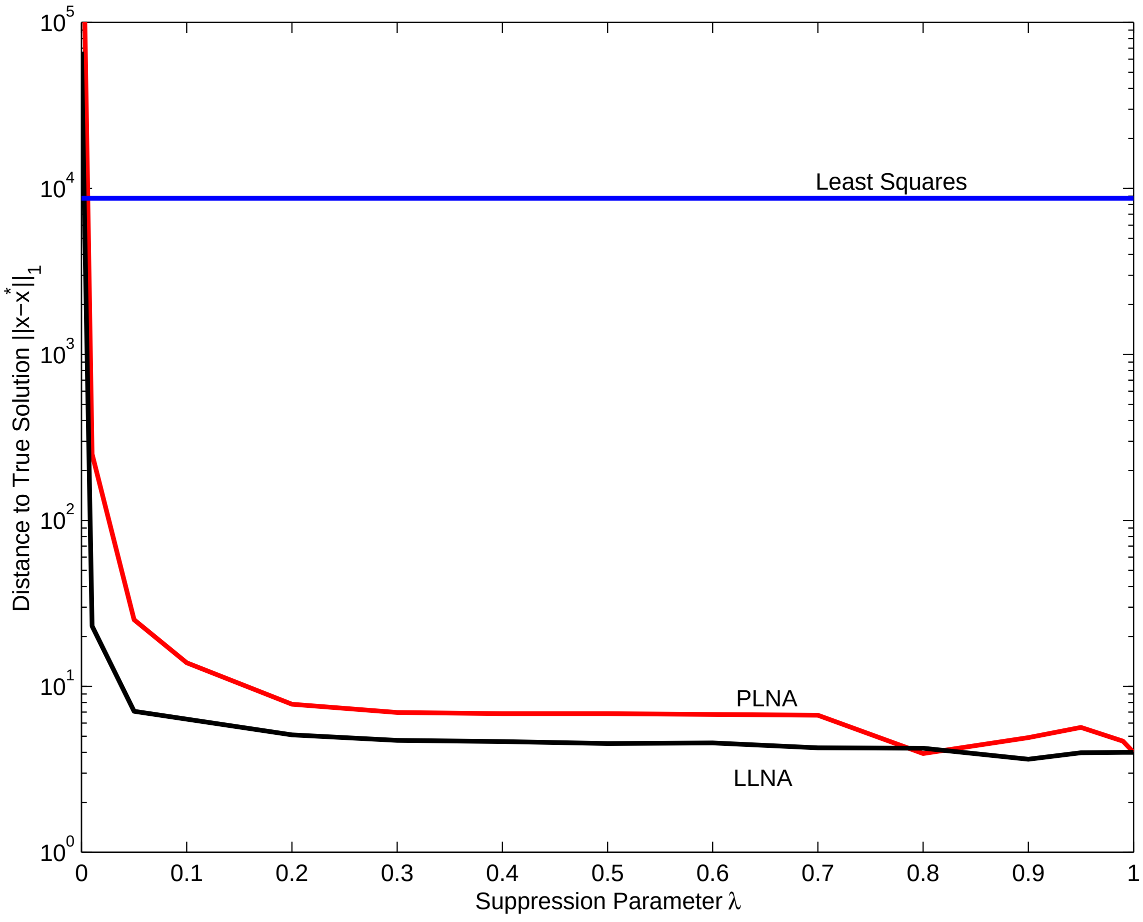
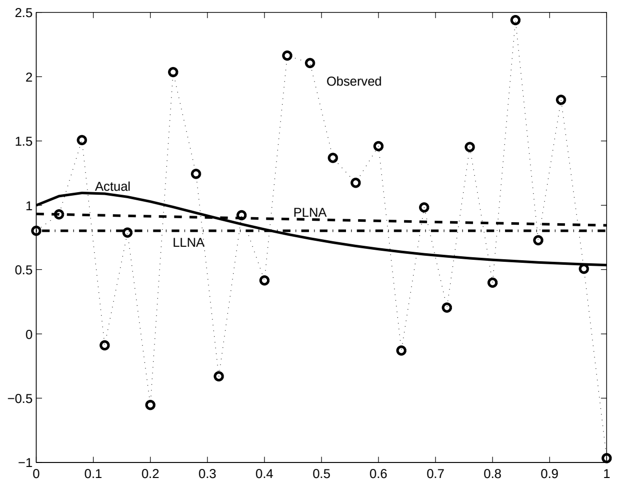
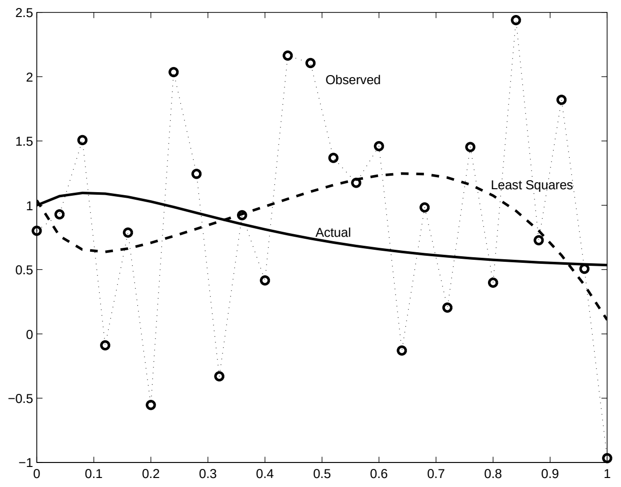


Figure 3: Average $\|x(\lambda) - x^*\|_1$ versus λ, where $x(\lambda)$ is a PLNA solution (25) in the curve marked PLNA and is an LLNA solution of (26) for the curve marked LLNA, compared with the average $\|x(ls) - x^*\|_1$, where $x(ls)$ is the least squares solution (27) solved by Algorithm 4.1. The true solution x^* (21) is such that $Ax^* = b$. The PLNA and LLNA solutions were computed for values of $\lambda = 0, 0.01, 0.05, 0.10, 0.20, \ldots, 0.90, 0.95, 0.99, 1$.



(a) Dashed curves are the recovered signal $\hat{g}(t)$ with coefficient vector $x(\lambda)$ determined by (25) with $\lambda = 0.3$ and $\|Ax(\lambda) - b\|_1 = 4.7410$ for PLNA and by (26) with $\lambda = 0.8$ and $\|Ax(\lambda) - b\|_1 = 4.7076$ for LLNA.



(b) Dashed curve is the recovered signal $\hat{g}(t)$ with coefficient vector $x(ls)$ determined by **least squares** solution (27) solved by Algorithm 4.1. Note: $\|Ax(ls) - b\|_1 = 8.9733$.

Figure 4: **Signal Recovery.** Solid curve is the true signal $g(t)$. Circles are the observed signal $\tilde{g}(t_i)$ sampled at discrete times and the dashed curves are the recovered signals.

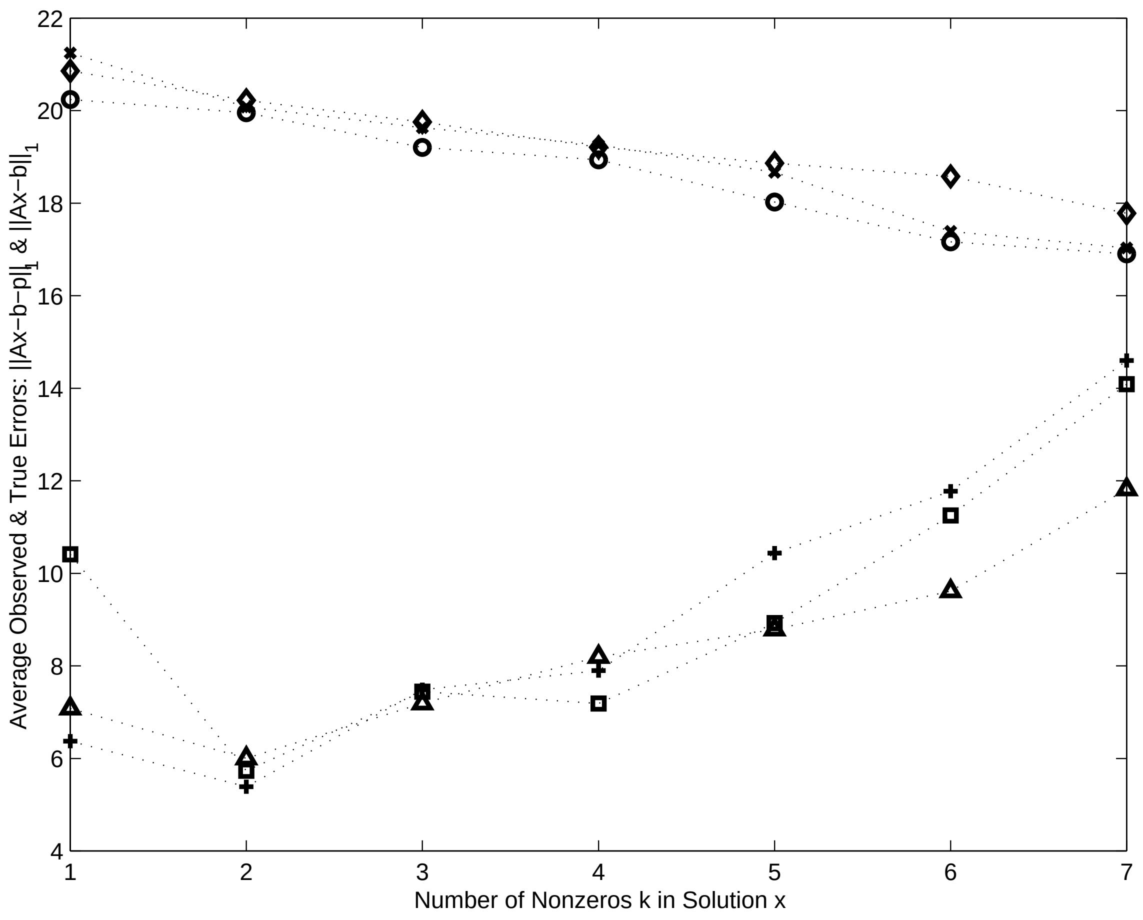


Figure 5: Comparison of PLNA (30) and LLNA (31) with combinatorial search (32). Average $\|Ax(\lambda) - b - p\|_1$ is '**x**' for PLNA and '$\diamond$' for LLNA. Average $\|Ax_c - b - p\|_1$ is '**o**'. Average $\|Ax(\lambda) - b\|_1$ is '$\square$' for PLNA and $\triangle$ for LLNA. Average $\|Ax_c - b\|_1$ is '$+$' for combinatorial solution x_c.

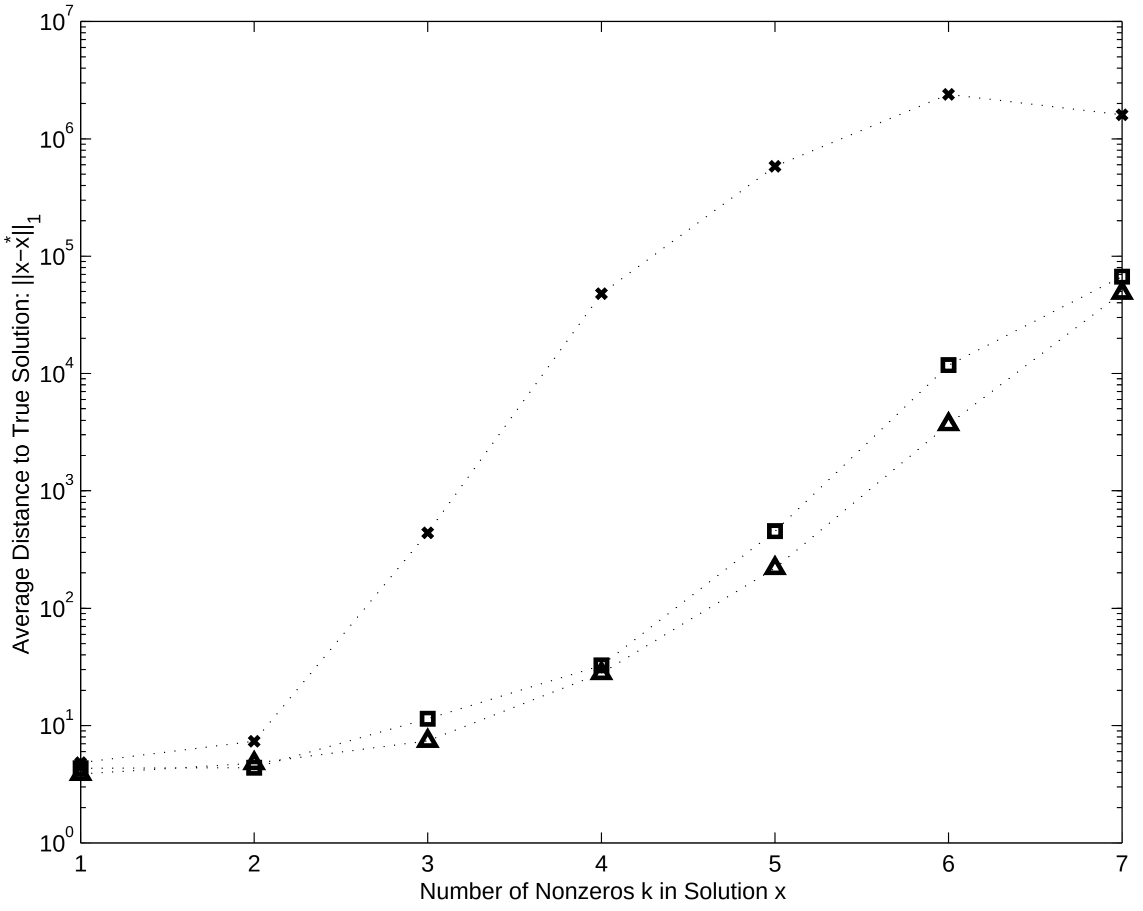
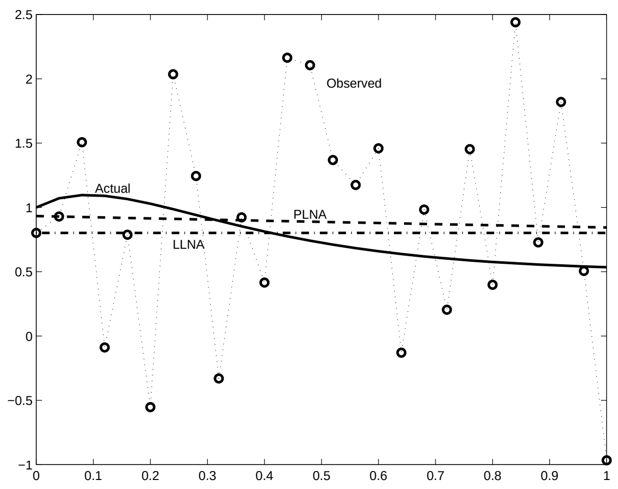
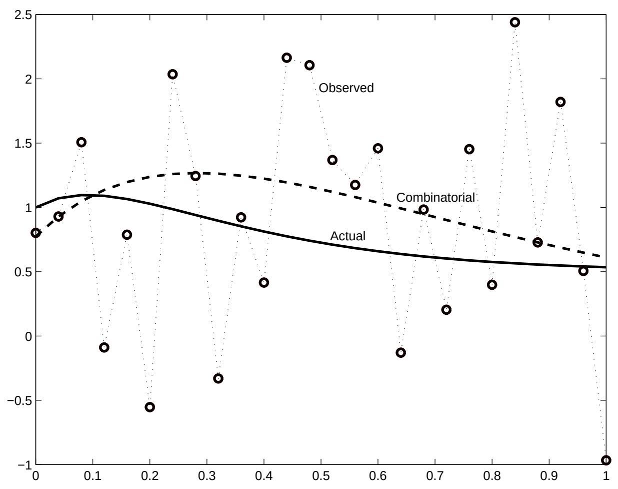


Figure 6: Comparison of PLNA (30) and LLNA (31) with combinatorial search (32). Average $\|x_c - x^*\|_1$ is '**x**'. Average $\|x(\lambda) - x^*\|_1$ is '□' for PLNA and $\triangle$ for LLNA. The true solution x^* is such that $Ax^* = b$.



(a) Dashed curves are the recovered signal $\hat{g}(t)$ with coefficient vector $x(\lambda)$ determined by (25) with $\lambda = 0.3$ and $\|Ax(\lambda) - b\|_1 = 4.7410$ for PLNA, and by (26) with $\lambda = 0.8$ and $\|Ax(\lambda) - b\|_1 = 4.7076$ for LLNA.



(b) Dashed curve is the recovered signal $\hat{g}(t)$ with coefficient vector x_c determined by **combinatorial search** with $k = 2$ (32). Note: $\|Ax_c - b\|_1 = 6.7826$.

Figure 7: **Signal Recovery.** Solid curve is the true signal $g(t)$. Circles are the observed signal $\tilde{g}(t_i)$ sampled at discrete times and the dashed curves are the recovered signals.

recovered by LLNA (26) for $\lambda = 0.8$. Figure 7(b) displays the true signal, the observed signal and signal recovered by combinatorial search solution x_c of (32) for $k = 2$.

4.3 Observations

We make the following observations with respect to the comparison between the PLNA, LLNA solutions and least squares solutions.

1. For all values of $\lambda \geq 0.05$ tested, the average observed system (training set) residual $\|Ax(ls) - b - p\|_1$ was strictly less than the average $\|Ax(\lambda) - b - p\|_1$ for both PLNA and LLNA. The least squares Algorithm 4.1 for solving (27) produced "better" solutions to the observed system $Ax = b + p$. See Figure 1. However......

2. For values of $\lambda \in [0.01, 0.90]$ tested, the PLNA average true system (testing set) residual $\|Ax(\lambda) - b\|_1$ was strictly less than the average $\|Ax(ls) - b\|_1$ indicating that PLNA produced "better" solutions to the true system $Ax = b$ in comparison with least squares. For values of $\lambda \in [0.01, 0.80]$ tested, the average true system residual with solutions determined by LLNA was also strictly less than the corresponding least squares true system residuals. See Figure 2. PLNA with $\lambda = 0.3$ and an average of 2.2 nonzero terms achieved an error reduction of 38.85% over the corresponding error obtained by the least squares solution. LLNA with

$\lambda = 0.8$ produced an average 1-norm true system residual that was 52.98% less than the least squares residual.

3. For values of $\lambda > 0.1$ tested, the average $\|x(\lambda) - x^*\|_1$, determined by both PLNA and LLNA, was 2 orders of magnitude less than the average $\|x(ls) - x^*\|$. Hence the PLNA and LLNA solutions were "closer" to recovering the true signal $g(t)$ (18). See Figure 3.

4. Figure 4, shows the most significant comparison between PLNA, LLNA and least squares: A much more accurate recovery of the true signal by both PLNA and LLNA than by least squares.

We note the following with respect to the comparison between the PLNA, LLNA solutions and the solutions obtained by combinatorial search.

1. For $k = 3, 4, 5, 6, 7$, the average PLNA $\|Ax(\lambda) - b\|_1$ was strictly less than the average $\|Ax_c - b\|_1$. For $k = 1, 2$, the average PLNA $\|Ax(\lambda) - b\|_1$ was less than or equal to 1.634 times the average $\|Ax_c - b\|_1$. For $k = 3, 5, 6, 7$, the average LLNA $\|Ax(\lambda) - b\|_1$ was strictly less than the corresponding average true system residual with the combinatorial solutions. For $k = 1, 2, 4$, the average LLNA $\|Ax(\lambda) - b\|_1$ was less than or equal to 1.114 times the corresponding average $\|Ax_c - b\|$. See Figure 5.

2. For $k \geq 3$, the average $\|x(\lambda) - x^*\|_1$, for both PLNA and LLNA, was strictly less than the average $\|x_c - x^*\|_1$ by orders of magnitude. For $k = 0, 1, 2$, average $\|x(\lambda) - x^*\|_1$ was less than or equal to average $\|x_c - x^*\|_1$. See Figure 6.

3. The minimum over $k = 1, \ldots, 7$ of the true system 1-norm residual of 5.3867 occurs for $k = 2$ with the solution obtained by combinatorial search. The true system residual for PLNA with $k = 2$ is 5.7330 and the true system residual for LLNA is 6.0022. We note that when computing the PLNA and LLNA solutions for $k = 2$, the first value of λ found (by a bisection search) such that the solution has 2 nonzero elements was chosen. This fact accounts for the discrepancy between the true system residuals in Figure 5 and Figure 2.

4. Figure 7 shows recovery of the true signal by both PLNA and LLNA which is as good or even better than the recovered signal by a lengthy combinatorial search.

The time needed by each approach to compute a solution was determined by performing a single run on a Sun SparcStation 20 with 96 megabytes of memory running MATLAB 5.1, using the commands "tic" and "toc" [21]. All linear programs were solved with CPLEX [7] interfaced with MATLAB. Solving the PLNA problem with $\lambda = 0.5$ with initial point (24) and $\alpha = 5$ took 0.4603 seconds. Solving the LLNA problem with $\lambda = 0.5$ took 0.1978 seconds. Determining the least squares solution by Algorithm 4.1 with $\tau = 0.0001$ took 0.0224 seconds. Determining the solution by combinatorial search with $k = 3$ took 13.2008 seconds.

Solutions computed by PLNA and LLNA were at most superior or at least comparable to those obtained by combinatorial search (32), yet needing two orders of magnitude less time to compute.

5 Conclusion

A theoretically justifiable fast finite algorithm has been proposed for solving linear systems corrupted by noise or errors in measurement. The parsimonious approach (PLNA) attempts to set to zero as many components of the solution vector as possible while minimizing the residual error of the corrupted system, whereas the least norm approach (LLNA) minimizes the norm of the solution as well as the residual. Numerical evidence indicates that both these two approaches lead to solutions with many zero components, and that such solutions may be closer by orders of magnitude to the solution of the underlying uncorrupted system than other solutions of the corrupted system obtained by either least squares or even by a time-consuming combinatorial search for a solution with a minimal number of nonzero components. It is interesting to note that parametricly minimizing the norm of the solution leads also to suppression of its components, and conversely parametrically suppressing components of the solution also leads to a solution with a reduced norm. Most importantly, PLNA and LLNA recover a much more accurate signal than that obtained by least squares and much faster than that obtained by a lengthy combinatorial search.

Acknowledgement

We are indebted to the referees for constructive comments that led to considerable improvements. In particular we are indebted to a referee who suggested the inclusion of the LLNA approach in our comparative tests.

References

[1] T. J. Abatzoglou, J. M. Mendel, and G. A. Harada. The constrained total least squares technique and its application to harmonic superposition. *IEEE Transactions on Signal Processing*, 39:1070–1087, 1991.

[2] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. Technical Report 96-15, Cornell University, Ithaca, NY, 1976. To Appear in Theoretical Computer Science. Available at http://www.cs.cornell.edu/Info/People/amaldi/amaldi.html.

[3] K. P. Bennett and J. A. Blue. A support vector machine approach to decision trees. Department of Mathematical Sciences Math Report No. 97-100, Rensselaer Polytechnic Institute, Troy, NY 12180, 1997. Available at http://www.math.rpi.edu/ bennek/.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[5] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 368–374, Cambridge, MA, 1997. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/96-03.ps.Z.

[6] P. S. Bradley, O. L. Mangasarian, and W. N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 1998. To appear. Available at ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-21.ps.Z.

[7] CPLEX Optimization Inc., Incline Village, Nevada. *Using the CPLEX(TM) Linear Optimizer and CPLEX(TM) Mixed Integer Optimizer (Version 2.0)*, 1992.

[8] Luc Dinh. *Theory of Vector Optimization.* Lecture Notes in Economics and Mathematical Systems 319. Springer Verlag, Berlin, 1989.

[9] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[10] Arthur A. Giordano. *Least Square Estimation With Applications to Digital Signal Processing.* John Wiley & Sons, New York, 1985.

[11] G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, 17:883–893, 1980.

[12] M. H. Hassoun. *Fundamentals of Artificial Neural Networks.* MIT Press, Cambridge, MA, 1995.

[13] S. Van Huffel, H. Park, and J. B. Rosen. Formulation and solution of structured total least norm problems for parameter estimation. *IEEE Transactions on Signal Processing*, 44:2464–2474, 1996.

[14] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem, Computational Aspects and Analysis.* SIAM, Philadelphia, PA, 1991.

[15] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice Hall, Englewood Cliffs, New Jersey, 1974.

[16] Y. le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II (Denver 1989)*, pages 598–605, San Mateo, California, 1990. Morgan Kaufmann.

[17] O. L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[18] O. L. Mangasarian. Machine learning via polyhedral concave minimization. In H. Fischer, B. Riedmueller, and S. Schaeffler, editors, *Applied Mathematics and Parallel Computing - Festschrift for Klaus Ritter*, pages 175–188. Physica-Verlag A Springer-Verlag Company, Heidelberg, 1996. Available at ftp://ftp.cs.wisc.edu/math-prog/tech-reports/95-20.ps.Z.

[19] O. L. Mangasarian. Mathematical programming in data mining. *Data Mining and Knowledge Discovery*, 1(2):183–201, 1997. Available at ftp://ftp.cs.wisc.edu/math-prog/tech-reports/96-05.ps.Z.

[20] O. L. Mangasarian. Solution of general linear complementarity problems via nondifferentiable concave minimization. *Acta Mathematica Vietnamica*, 22(1):199–205, 1997. Available at ftp://ftp.cs.wisc.edu/math-prog/tech-reports/96-10.ps.Z.

[21] MATLAB. *User's Guide.* The MathWorks, Inc., 1992.

[22] B. T. Polyak. *Introduction to Optimization.* Optimization Software, Inc., Publications Division, New York, 1987.

[23] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, New Jersey, 1970.

[24] J. Ben Rosen, Haesun Park, and John Glick. Total least norm formulation and solution for structured problems. *SIAM Journal on Matrix Analysis*, 17(1):110–128, January 1996.

[25] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10:153–178, 1993.

[26] J. W. Shavlik and T. G. Dietterich (editors). *Readings in Machine Learning.* Morgan Kaufman, San Mateo, California, 1990.

[27] G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, Wellesley, MA, 1993.

[28] Charles W. Therrien. *Discrete Random Signals and Statistical Signal Processing.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1992.

[29] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[30] D. H. Wolpert, editor. *The Mathematics of Generalization*, Reading, MA, 1995. Addison-Wesley.