Sparse pseudoinverses via LP and SDP relaxations of Moore-Penrose*

Victor K. Fuentes
University of Michigan
vicfuen@umich.edu

Marcia Fampa
Universidade Federal do Rio de Janeiro
fampa@cos.ufrj.br

Jon Lee
University of Michigan
jonxlee@umich.edu

June 15, 2016

Abstract

Pseudoinverses are ubiquitous tools for handling over- and under-determined systems of equations. For computational efficiency, sparse pseudoinverses are desirable. Recently, sparse left and right pseudoinverses were introduced, using 1-norm minimization and linear programming. We introduce several new sparse pseudoinverses by developing linear and semi-definite programming relaxations of the well-known Moore-Penrose properties.

Keywords: sparse; pseudoinverse; semi-definite programming; linear programming.

Introduction

Pseudoinverses are a central tool in matrix algebra and its applications. Sparse optimization is concerned with finding sparse solutions of optimization problems, often for computational efficiency in the use of the output of the optimization. There is usually a tradeoff between an ideal dense solution and a less-ideal sparse solution, and sparse optimization is often focused on tractable methods for striking a good balance. Recently, sparse optimization has been used to calculate tractable sparse left and right pseudoinverses, via linear programming. We extend this theme to derive several other tractable sparse pseudoinverses, employing linear and semi-definite programming.

In §1, we give a very brief overview of pseudoinverses, and in §2, we describe some prior work on sparse left and right pseudoinverses. In §3, we present new sparse pseudoinverses based on tractable

*Partial support from NSF grant CMMI–1160915 and ONR grant N00014-14-1-0315.

convex relaxations of the Moore-Penrose properties. In §4, we present preliminary computational results. Finally, in §5, we make brief conclusions and describe our ongoing work.

In what follows, $I_n/\mathbf{0}_n/\vec{0}_n/\vec{1}_n$ denotes an order-n identity matrix/zero matrix/zero vector/all-ones vector.

1 Pseudoinverses

When a real matrix $A \in \mathbb{R}^{m\times n}$ is not square or not invertible, we consider pseudoinverses of A (see [11] for a wealth of information on this topic). For example, there is the well-known *Drazin inverse* for square and even non-square matrices (see [3]) and the *generalized Bott-Duffin inverse* (see [2]).

The most well-known pseudoinverse of all is the *M-P (Moore-Penrose) pseudoinverse*, independently discovered by E.H. Moore and R. Penrose. If $A = U\Sigma V'$ is the real singular value decomposition of A (see [5], for example), then the M-P pseudoinverse of A can be defined as $A^+ := V\Sigma^+U'$, where Σ^+ has the shape of the transpose of the diagonal matrix Σ, and is derived from Σ by taking reciprocals of the non-zero (diagonal) elements of Σ (i.e., the non-zero singular values of A). The M-P pseudoinverse, a central object in matrix theory, has many concrete uses. For example, we can use it to solve least-squares problems, and we can use it, together with a norm, to define condition numbers of matrices. The M-P pseudoinverse is calculated, via its connection with the real singular value decomposition, by the `Matlab` function `pinv`.

2 Sparse left and right pseudoinverses

It is well known that in the context of seeking a sparse solution in a convex set, a surrogate for minimizing the sparsity is to minimize the 1-norm. In fact, if the components of the solution have absolute value no more than unity, a minimum 1-norm solution has 1-norm no greater than the number of nonzeros in the sparsest solution. With this in mind, [4] defines sparse left and right pseudoinverses in a natural and tractable manner. Below, $\|\cdot\|_1$ denotes entry-wise 1-norm.

For an "overdetermined case", [4] defines a *sparse left pseudoinverse* via the convex formulation

$$\min\left\{\|H\|_1 \;:\; HA = I_n\right\}. \tag{$\mathcal{O}$}$$

For an "underdetermined case", [4] defines a *sparse right pseudoinverse* via the convex formulation

$$\min\left\{\|H\|_1 \;:\; AH = I_m\right\}. \tag{$\mathcal{U}$}$$

These definitions emphasize sparsity, while in some sense putting a rather mild emphasis on the aspect of being a pseudoinverse. We do note that if the columns of A are linearly independent, then the M-P pseudoinverse is precisely $(A'A)^{-1}A'$, which is a left inverse of A. Therefore, if A has full column rank, then the M-P pseudoinverse is a feasible H for $(\mathcal{O})$. Conversely, if A does not have full column rank, then $(\mathcal{O})$ has no feasible solution, and so there is no sparse left inverse in such a case. On the other hand, if the rows of A are linearly independent, then the M-P pseudoinverse is precisely $A'(AA')^{-1}$, which is a right inverse of A. Therefore, if A has full row rank, then the M-P pseudoinverse is a feasible H for $(\mathcal{U})$. Conversely, if A does not have full row rank, then $(\mathcal{U})$ has no feasible solution, and so there is no sparse right inverse in such a case.

These sparse pseudoinverses are easy to calculate, by linear programming:

$$\min\left\{\sum_{ij\in m\times n} t_{ij} \;:\; t_{ij}\geq h_{ij},\; t_{ij}\geq -h_{ij},\; \forall ij\in m\times n;\; HA=I_n\right\} \tag{$LP_{\mathcal{O}}$}$$

for the sparse left pseudoinverse, and

$$\min\left\{\sum_{ij\in m\times n} t_{ij} \;:\; t_{ij}\geq h_{ij},\; t_{ij}\geq -h_{ij},\; \forall ij\in m\times n;\; AH=I_m\right\} \tag{$LP_{\mathcal{U}}$}$$

for the sparse right pseudoinverse. In fact, the ($LP_{\mathcal{O}}$) decomposes row-wise for H, and ($LP_{\mathcal{U}}$) decomposes column-wise for H, so calculating these sparse pseudoinverses can be made very efficient at large scale. Also, these sparse pseudoinverses do have nice mathematical properties (see [4]).

3 New sparse relaxed Moore-Penrose pseudoinverses

We seek to define different tractable sparse pseudoinverses, based on the the following nice characterization of the M-P pseudoinverse.

Theorem 3.1. *For $A\in R^{m\times n}$, the M-P pseudoinverse A^+ is the unique $H\in\mathbb{R}^{n\times m}$ satisfying:*

$$AHA = A \tag{P1}$$
$$HAH = H \tag{P2}$$
$$(AH)' = AH \tag{P3}$$
$$(HA)' = HA \tag{P4}$$

If we consider properties (P1) - (P4) which characterize the M-P pseudoinverse, we can observe that properties (P1), (P3) and (P4) are all linear in H, and the only non-linearity is property (P2), which is quadratic. Another important point to observe is that without property (P1), H could be the all-zero matrix and satisfy properties (P2), (P3) and (P4). Whenever property (P1) holds, H is called a *generalized inverse*. So, in the simplest approach, we can consider minimizing $\|H\|_1$ subject to property (P1) and *any subset* of the properties (P3) and (P4). In this manner, we get several (four) new sparse pseudoinverses which can all be calculated by linear programming.

To go further, we can also consider convex relaxations of property (P2). To pursue that direction, we enter the realm of semi-definite programming (see [1],[16],[17], for example).

We can see property (P2) as

$$h_{i\cdot}Ah_{\cdot j} = h_{ij},$$

for all $ij\in m\times n$. So, we have mn quadratic equations to enforce, which we can see as

$$\frac{1}{2}\left(h_{i\cdot}, h'_{\cdot j}\right)\begin{bmatrix}\mathbf{0}_m & A\\ A' & \mathbf{0}_n\end{bmatrix}\begin{pmatrix}h'_{i\cdot}\\ h_{\cdot j}\end{pmatrix} = h_{ij}, \tag{3.1}$$

for all $ij\in m\times n$. We can view these quadratic equations (3.1) as

$$\frac{1}{2}\left\langle Q, \begin{pmatrix}h'_{i\cdot}\\ h_{\cdot j}\end{pmatrix}\left(h_{i\cdot}, h'_{\cdot j}\right)\right\rangle = h_{ij},$$

for all $ij \in m \times n$, where

$$Q := \begin{bmatrix} \mathbf{0}_m & A \\ A' & \mathbf{0}_n \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)},$$

and $\langle \cdot, \cdot \rangle$ denotes element-wise dot-product.

Now, we lift the variables to matrix space, defining matrix variables

$$\mathcal{H}_{ij} := \begin{pmatrix} h'_{i\cdot} \\ h_{\cdot j} \end{pmatrix} \left(h_{i\cdot}, h'_{\cdot j}\right) \in \mathbb{R}^{(m+n)\times(m+n)},$$

for all $ij \in m \times n$. So, we can see (3.1) as the *linear* equations

$$\frac{1}{2} \langle Q, \mathcal{H}_{ij} \rangle = h_{ij}, \tag{3.2}$$

for all $ij \in m \times n$, together with the *non-convex* equations

$$\mathcal{H}_{ij} - \begin{pmatrix} h'_{i\cdot} \\ h_{\cdot j} \end{pmatrix} \left(h_{i\cdot}, h'_{\cdot j}\right) = \mathbf{0}_{m+n}, \tag{3.3}$$

for all $ij \in m \times n$. Next, we relax the equations (3.3) via the *convex semi-definiteness* constraints:

$$\mathcal{H}_{ij} - \begin{pmatrix} h'_{i\cdot} \\ h_{\cdot j} \end{pmatrix} \left(h_{i\cdot}, h'_{\cdot j}\right) \succeq \mathbf{0}_{m+n}, \tag{3.4}$$

for all $ij \in m \times n$. So we can relax the M-P property (P2) as (3.2) and (3.4), for all $ij \in m \times n$.

To put (3.4) into a standard form for semi-definite programming, we create variables vectors $x_{ij} \in \mathbb{R}^{m+n}$, and we have linear equations

$$x_{ij} = \begin{pmatrix} h'_{i\cdot} \\ h_{\cdot j} \end{pmatrix}. \tag{3.5}$$

Next, for all $ij \in m \times n$, we introduce symmetric positive semi-definite matrix variables $Z_{ij} \in \mathbb{R}^{(m+n+1)\times(m+n+1)}$, interpreting the entries as follows:

$$Z_{ij} = \begin{bmatrix} x_{ij}^{(0)} & x'_{ij} \\ x_{ij} & \mathcal{H}_{ij} \end{bmatrix}. \tag{3.6}$$

Then the linear equation

$$x_{ij}^{(0)} = 1 \tag{3.7}$$

and $Z_{ij} \succeq \mathbf{0}_{(m+n+1)\times(m+n+1)}$ precisely enforce (3.4).

Finally, we re-cast (3.2) as

$$\frac{1}{2} \left\langle \bar{Q}, Z_{ij} \right\rangle = h_{ij}, \tag{3.8}$$

where

$$\bar{Q} := \begin{bmatrix} 0 & \vec{0}'_{m+n} \\ \vec{0}_{m+n} & Q \end{bmatrix} \in \mathbb{R}^{(m+n+1)\times(m+n+1)}. \tag{3.9}$$

In summary, we can consider minimizing $\|H\|_1$ subject to property (P1) and *any subset* of (P3), (P4), and (3.2)+(3.4) for all $ij \in m \times n$ (though we reformulate (3.2)+(3.4) as above, so it is in a convenient form for semi-definite programming solvers like `CVX`). In doing so, we get many (eight) new sparse pseudoinverses which are all tractable (via linear or semi-definite programming).

Of course all of these optimization problems have feasible solutions, because the M-P pseudoinverse A^+ always gives a feasible solution. For the cases in which we have semi-definite programs, an important issue is whether there is a strictly feasible solution — the *Slater condition(/constraint qualification)* — as that is sufficient for strong duality to hold and affects the convergence of algorithms (e.g., see [1]). Even if the Slater condition does not hold, there is a facial-reduction algorithm that can induce the Slater condition to hold on an appropriate face of the feasible region (see [10]).

4 Preliminary computational experiments

We made some preliminary tests of our ideas, using CVX/Matlab (see [7], [6]). Before describing our experimental setup, we observe the following results.

Proposition 4.1. *If A has full column rank n and H satisfies (P1), then H is a left inverse of A, and H satisfies (P2) and (P4). If A has full row rank m and H satisfies (P1), then H is a right inverse of A, and H satisfies (P2) and (P3),*

Corollary 4.2. *If A has full column rank n and H satisfies (P1) and (P3), then $H = A^+$. If A has full row rank m and H satisfies (P1) and (P4), then $H = A^+$.*

Because of these results, we decided to focus our experiments on matrices A with rank less than $\min\{m, n\}$. We generated random dense $n \times n$ rank-r matrices A of the form $A = UV$, where each U and V' are $n \times r$, with $n = 40$, and five instance for each $r = 4, 8, 16, \ldots, 36$. The entries in U and V were iid uniform $(-1, 1)$. We then scaled each A by a multiplicative factor of 0.01, which had the effect of making $A+$ fully dense to an entry-wise zero-tolerance of 0.1. In computing various sparse pseudoinverses, we used a zero-tolerance of 10^{-5}. We measured sparsity of a sparse pseudoinverse as the number of its nonzero components divided by n^2. We measured quality of a sparse pseudoinverse H, relative to the M-P pseudoinverse A^+ in two ways:

- *least-squares ratio* ('lsr'): $\|AHb - b\|_2 / \|AA^+b - b\|_2$, with arbitrarily $b := \vec{1}_m$. (Note that $x := A^+b$ always minimizes $\|Ax - b\|_2$.)
- 2-*norm ratio* ('2nr'): $\|HA\vec{1}_n\|_2 / \|A^+A\vec{1}_n\|_2$. (Note that $x := HA\vec{1}_n$ is always a solution to $Ax = A\vec{1}_n$, whenever H satisfies (P1), and one that minimizes $\|x\|_2$ is given by $x := A^+A\vec{1}_n$.)

Proposition 4.3. *If H satisfies (P1) and (P3), then $AH = AA^+$.*

Proof.

$$\begin{aligned} AHA &= AA^+A && \text{(by (P1))} \\ H'A'A &= (A^+)'A'A && \text{(by (P3))} \\ A'AH &= A'AA^+ \\ (A^+)'A'AH &= (A^+)'A'AA^+ \\ AH &= AA^+, \end{aligned}$$

the last equation following directly from a well-known property of A^+. □

Corollary 4.4. *If H satisfies (P1) and (P3), then $x := Hb$ (and of course A^+b) solves* $\min\{\|Ax - b\|_2 \ : \ x \in \mathbb{R}^n\}$.

Similarly, we have the following two results:

Proposition 4.5. *If H satisfies (P1) and (P4), then $HA = A^+A$.*

Corollary 4.6. *If H satisfies (P1) and (P4), and b is in the column space of A, then Hb (and of course A^+b) solves* $\min\{\|x\|_2 \ : \ Ax = b, \ x \in \mathbb{R}^n\}$.

So in the situations covered by Corollaries 4.4 and 4.6, we can seek and use sparser pseudoinverses than A^+. Our computational results are summarized in Table 1. '1nr' (1-norm ratio) is simply $\|H\|_1/\|A^+\|_1$. 'sr' (sparsity ratio) is simply $\|H\|_0/\|A^+\|_0$. Note that the entries of 1 reflect the results above. We observe that sparsity can be gained versus the M-P pseudoinverse, often with a modest decrease in quality of the pseudoinverse, and we can observe some trends as the rank varies.

5 Conclusions and ongoing work

We have introduced eight tractable pseudoinverses based on using 1-norm minimization to induce sparsity and making convex relaxations of the M-P properties. It remains to be seen if any of these new pseudoinverses will be found to be valuable in practice. There is a natural tradeoff between sparsity and closeness to the M-P properties, and where one wants to be on this spectrum may well be application dependent. We are in the process of carrying out more thorough experiments.

In particular, we are testing our sparse pseudoinverses that need semi-definite programming. In the manner of [13] and [14], we may go further and enforce more of (3.3) using "disjunctive cuts", producing a better convex relaxation of (3.3) than (3.4). But this would come at some significant costs: (i) greater computational effort, (ii) decreased sparsity as we work our way toward enforcing more of the M-P properties, (iii) lack of specificity.

Another idea that we are exploring is to develop update algorithms for sparse pseudoinverses. Of course the Sherman-Morrison-Woodbury formula gives us a convenient way to update a matrix inverse of A after a low-rank modification. Extending that formula, A^+ can be updated efficiently (see [8] and [12]). It is an interesting challenge to see if we can take advantage of a sparse pseudoinverse of A in calculating a sparse pseudoinverse of a low-rank modification of A.

In related work, we are in the process of investigating techniques for decomposing an input matrix $\bar{C}$ into $A + B$, where B has low rank and A has a sparse (pseudo)inverse. In that context, A is a matrix *variable*, so even the left- and right-inverse constraints ($HA = I_n$ and $AH = I_m$) are non-convex quadratic equations. So, already in that context, we are applying similar ideas to the ones we presented here for relaxing these equations. When we instead consider our new sparse pseudoinverses (based on the M-P properties), again in the context where A is a matrix variable, already the M-P properties (P3) and (P4) are quadratic and properties (P1) and (P2) are cubic (in A and H). We can still apply our basic approach for handling quadratic equations, now to (P3) and (P4). As for (P1) and (P2), we can take a variety of approaches. One possibility is to introduce auxiliary scalar variables to get back to quadratic equations; though even then there are issues to consider in choosing the best way to carry this out (see [15]). Another possibility is to introduce auxiliary *matrix* variables, to get back to quadratic matrix equations — some fascinating questions then arise if we consider how to extend the results in [15]. Another possibility is to employ semi-definite programming relaxations for polynomial systems (e.g., see [9]).

Table 1: Sparsity vs quality ($m = n = 40$)

		P1				P1+P3				P1+P4				P1+P3+P4			
r	$\|A^+\|_1$	1nr	sr	lsr	2nr	1nr	sr	lsr	2nr	1nr	sr	lsr	2nr	1nr	sr	lsr	2nr
4	586	0.44	0.01	1.07	2.27	0.60	0.10	1	1.43	0.64	0.10	1.02	1	0.75	0.19	1	1
4	465	0.46	0.01	1.07	1.82	0.63	0.10	1	1.43	0.63	0.10	1.01	1	0.77	0.19	1	1
4	500	0.44	0.01	1.08	1.82	0.62	0.10	1	1.46	0.62	0.10	1.01	1	0.76	0.19	1	1
4	503	0.41	0.01	1.28	2.00	0.62	0.10	1	1.31	0.62	0.10	1.06	1	0.75	0.19	1	1
4	511	0.45	0.01	1.10	2.36	0.63	0.10	1	1.55	0.64	0.10	1.09	1	0.78	0.19	1	1
8	855	0.53	0.04	1.17	1.63	0.69	0.20	1	1.28	0.68	0.20	1.05	1	0.80	0.36	1	1
8	851	0.53	0.04	1.22	1.60	0.69	0.20	1	1.33	0.69	0.20	1.07	1	0.80	0.36	1	1
8	841	0.53	0.04	1.25	1.70	0.69	0.20	1	1.34	0.69	0.20	1.07	1	0.80	0.36	1	1
8	761	0.52	0.04	1.05	1.70	0.69	0.20	1	1.32	0.68	0.20	1.09	1	0.81	0.36	1	1
8	864	0.52	0.04	1.09	1.40	0.69	0.20	1	1.21	0.68	0.20	1.04	1	0.80	0.36	1	1
12	1150	0.60	0.09	1.26	1.68	0.74	0.30	1	1.26	0.75	0.30	1.12	1	0.86	0.51	1	1
12	1198	0.59	0.09	1.20	1.70	0.75	0.30	1	1.25	0.75	0.30	1.05	1	0.85	0.51	1	1
12	1236	0.59	0.09	1.10	1.28	0.75	0.30	1	1.17	0.75	0.30	1.24	1	0.86	0.51	1	1
12	1134	0.60	0.09	1.38	1.43	0.75	0.30	1	1.19	0.74	0.30	1.09	1	0.85	0.51	1	1
12	1135	0.60	0.09	1.20	1.44	0.75	0.30	1	1.21	0.75	0.30	1.14	1	0.85	0.51	1	1
16	1643	0.67	0.16	1.36	1.85	0.79	0.40	1	1.30	0.80	0.40	1.17	1	0.90	0.64	1	1
16	1421	0.65	0.16	1.20	1.61	0.79	0.40	1	1.29	0.79	0.40	1.31	1	0.90	0.64	1	1
16	1518	0.65	0.16	1.33	1.38	0.79	0.40	1	1.20	0.80	0.40	1.30	1	0.89	0.64	1	1
16	1512	0.66	0.16	1.45	1.68	0.80	0.40	1	1.34	0.79	0.40	1.16	1	0.89	0.64	1	1
16	1539	0.65	0.16	1.18	1.25	0.79	0.40	1	1.19	0.79	0.40	1.29	1	0.89	0.64	1	1
20	2147	0.72	0.25	1.51	1.33	0.84	0.50	1	1.15	0.84	0.50	1.42	1	0.94	0.75	1	1
20	2111	0.72	0.25	1.81	1.44	0.83	0.50	1	1.35	0.84	0.50	1.48	1	0.93	0.75	1	1
20	2148	0.71	0.25	2.08	1.49	0.84	0.50	1	1.32	0.83	0.50	1.45	1	0.93	0.75	1	1
20	2061	0.72	0.25	1.50	1.49	0.84	0.50	1	1.35	0.84	0.50	1.31	1	0.93	0.75	1	1
20	2283	0.72	0.25	1.61	1.47	0.83	0.50	1	1.47	0.84	0.50	1.20	1	0.94	0.75	1	1
24	2865	0.77	0.36	1.86	1.24	0.87	0.60	1	1.18	0.87	0.60	1.51	1	0.96	0.84	1	1
24	3228	0.78	0.36	2.17	1.34	0.87	0.60	1	1.37	0.88	0.60	1.90	1	0.96	0.84	1	1
24	2884	0.77	0.36	2.27	1.72	0.87	0.60	1	1.32	0.87	0.60	1.55	1	0.96	0.84	1	1
24	2853	0.78	0.36	1.50	1.66	0.88	0.60	1	1.50	0.87	0.60	1.53	1	0.96	0.84	1	1
24	2944	0.78	0.36	1.72	1.48	0.87	0.60	1	1.64	0.88	0.60	1.64	1	0.96	0.84	1	1
28	4359	0.82	0.49	1.69	1.65	0.90	0.70	1	1.63	0.91	0.70	1.89	1	0.98	0.91	1	1
28	4268	0.83	0.49	2.27	1.98	0.91	0.70	1	1.79	0.91	0.70	2.08	1	0.98	0.91	1	1
28	4069	0.83	0.49	2.35	1.51	0.91	0.70	1	1.43	0.91	0.70	2.25	1	0.98	0.91	1	1
28	3993	0.83	0.49	2.30	1.58	0.90	0.70	1	1.27	0.91	0.70	2.19	1	0.97	0.91	1	1
28	4387	0.83	0.49	2.54	1.78	0.91	0.70	1	1.34	0.91	0.70	2.76	1	0.98	0.91	1	1
32	6988	0.88	0.64	4.08	1.60	0.94	0.80	1	1.81	0.94	0.80	3.54	1	0.99	0.96	1	1
32	6493	0.89	0.64	3.00	1.75	0.94	0.80	1	1.79	0.94	0.80	2.35	1	0.99	0.96	1	1
32	11445	0.89	0.64	4.50	4.82	0.94	0.80	1	2.58	0.94	0.80	7.18	1	0.99	0.96	1	1
32	8279	0.89	0.64	5.08	2.72	0.95	0.80	1	2.31	0.94	0.80	3.39	1	0.99	0.96	1	1
32	5069	0.89	0.64	2.14	1.90	0.95	0.80	1	1.74	0.94	0.80	2.26	1	0.99	0.96	1	1
36	18532	0.94	0.81	11.16	2.88	0.97	0.90	1	1.85	0.97	0.90	9.80	1	1.00	0.99	1	1
36	16646	0.94	0.81	10.91	2.53	0.97	0.90	1	3.04	0.97	0.90	8.07	1	1.00	0.99	1	1
36	11216	0.95	0.81	4.56	1.50	0.97	0.90	1	1.60	0.97	0.90	4.93	1	1.00	0.99	1	1
36	10299	0.95	0.81	6.12	1.45	0.98	0.90	1	2.14	0.97	0.90	5.37	1	1.00	0.99	1	1
36	11605	0.94	0.81	5.70	1.56	0.97	0.90	1	2.17	0.98	0.90	5.65	1	1.00	0.99	1	1

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[2] Y. Chen. The generalized Bott-Duffin inverse and its applications. *Linear Algebra and its Applications*, 134:71–91, 1990.

[3] R.E. Cline and T.N.E. Greville. A Drazin inverse for rectangular matrices. *Linear Algebra and its Applications*, 29:53–62, 1980.

[4] I. Dokmanić, M. Kolundžija, and M. Vetterli. Beyond Moore-Penrose: Sparse pseudoinverse. In *ICASSP 2013,* pp. 6526–6530. 2013.

[5] G.H. Golub and C.F. Van Loan. *Matrix Computations (3rd Ed.).* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[6] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control.* pp. 95–110. Springer, 2008.

[7] M. Grant and S. Boyd. CVX, version 2.1, 2015.

[8] C.D. Meyer. Generalized inversion of modified matrices. *SIAM J. App. Math*, 24:315–323, 1973.

[9] P.A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Prog.*, 96(2):293–320, 2003.

[10] G. Pataki. Strong duality in conic linear programming: Facial reduction and extended duals. In *Computational and Analytical Mathematics.* pp. 613–634. Springer, 2013.

[11] C.R. Rao and S.K. Mitra. *Generalized Inverse of Matrices and Its Applications.* Probability and Statistics Series. Wiley, 1971.

[12] K.S. Riedel. A Sherman-Morrison-Woodbury identity for rank augmenting matrices with application to centering. *SIAM J. Matrix Anal. Appl.*, 13(2):659–662, 1992.

[13] A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations. *Math. Prog., Ser. B*, 124:383–411, 2010.

[14] A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations. *Math. Prog., Ser. A*, 130:359–413, 2010.

[15] E. Speakman and J. Lee. Quantifying double McCormick. arXiv:1508.02966v2, 2015.

[16] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[17] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications.* Kluwer, Boston, 2000.