Refinement of approximate solution to conic LP

1 Introduction

Recently, solvers for large-scale linear conic optimization based on first-order methods have been developed. Compared to interior-point methods, these solvers have lower memory requirements and therefore scale better to large problem instances. Unfortunately, the scalability of first-order methods comes at a cost: computing solutions with high accuracy can often be expensive due to slow tail convergence. As a result, first-order methods such as SCS [4] sometimes terminate with a non-accurate solution. As an attempt to achieve higher accuracy, Anderson acceleration has recently been introduced in SCS [6]. Despite this effort, SCS can still sometimes terminate with a non-accurate solution.

This note acts as a tutorial on how to *refine* or *polish* an approximate solution for large-scale linear conic programs. The technique for refining an approximate solution has already been described in [2]. Hence, our main focus here is to present computational experiments illustrating the empirical performance of the refinement algorithm.

The outline of this note is as follows. In Section 2 we discuss the essential idea underlying the refinement algorithm. In Section 3 we present an open-source Python package, written in C++ with an interface to CVXPY [3, 1] that implements the refinement algorithm. In Section 4 the empirical performance of the refinement algorithm is illustrated.

2 Background

In this section we present the background needed for understanding the essential idea of the refinement algorithm. Further details can be found in [2]. Throughout this note we consider a primal-dual conic linear program of the form

$$\begin{array}{rlrl} \text{minimize} & c^T x & \text{minimize} & b^T y \\ \text{subject to} & Ax + s = b & \text{subject to} & A^T y + c = 0 \\ & (x, s) \in \mathbf{R}^n \times \mathcal{K}, & & y \in \mathcal{K}^*. \end{array} \tag{1}$$

Here $x \in \mathbf{R}^n$ and $s \in \mathbf{R}^m$ (with $n \leq m$) are referred to as *primal variables*, and $y \in \mathbf{R}^m$ is referred to as the *dual variable*. The set $\mathcal{K} \subseteq \mathbf{R}^m$ is a nonempty closed convex cone and

$\mathcal{K}^* \subseteq \mathbf{R}^m$ is its dual cone. The matrix $A \in \mathbf{R}^{m\times n}$, the cone $\mathcal{K}$ and the vectors $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$ are the *problem data.*

Weak duality for the primal-dual pair (1) states that for any primal feasible (x, s) and dual feasible y it holds that $c^T x \geq -b^T y$. Hence, any (x, s) satisfying $Ax + s = 0$, $s \in \mathcal{K}$, $c^T x = -1$ serves as a certificate that the dual is infeasible. Similarily, y satisfying $A^T y = 0$, $y \in \mathcal{K}^*$, $b^T y = -1$ serves as a certificate that the primal is infeasible.

2.1 Self-dual homogeneous embedding

If a point $(x, y, s) \in \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^m$ satisfies the KKT-conditions

$$\begin{bmatrix} 0 & A^T \\ -A & 0 \\ -c^T & -b^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ b \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ s \\ 0 \end{bmatrix}, \qquad x \in \mathbf{R}^n,\ y \in \mathcal{K}^*,\ s \in \mathcal{K}, \tag{2}$$

then (x, s) is primal optimal and y is dual optimal. If the conic linear program is an ordinary linear program, *i.e.,* $\mathcal{K} = \mathbf{R}^n_+$, then these conditions are also necessary for primal-dual optimality. Under a suitable constraint qualification (which is satisfied for most problems arising in everyday life), these conditions are also necessary for primal-dual optimality in the presence of more general cones such as the second-order or positive semidefinite cone.

A desirable feature in a general-purpose optimization solver is the ability to detect infeasibility or unboundedness. Designing a solver with the ability to detect infeasibility and unboundedness can be done by using the so-called *self-dual homogeneous embedding* for the primal-dual pair (1) [5]. In this embedding we augment the KKT-system (2) with two scalar variables τ, κ to obtain the system

$$\begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \tau \end{bmatrix} = \begin{bmatrix} 0 \\ s \\ \kappa \end{bmatrix}, \qquad x \in \mathbf{R}^n,\ y \in \mathcal{K}^*,\ \tau \geq 0,\ s \in \mathcal{K},\ \kappa \geq 0. \tag{3}$$

Note that the two new variables are complementary, *i.e.,* $\tau\kappa = 0$.

Let $(\bar{x}, \bar{y}, \bar{s}, \bar{\tau}, \bar{\kappa})$ denote a solution of (3). The values of $\bar{\tau}$ and $\bar{\kappa}$ encode different possible outcomes for the primal-dual pair (1). We have the following cases.

1. If $\bar{\tau} > 0$, the scaled point $(x, y, s) = \frac{1}{\bar{\tau}}(\bar{x}, \bar{y}, \bar{s})$ satisfies the KKT-conditions (2), so the scaled point is primal-dual optimal.
2. If $\bar{\kappa} > 0$ at least one of the quantities $c^T\bar{x}$ or $b^T\bar{y}$ is negative (since $-c^T\bar{x} - b^T\bar{y} = \bar{\kappa}$).
   (a) If $b^T\bar{y} < 0$, then $y = -\bar{y}/b^T\bar{y} \in \mathcal{K}$ satisfies $A^T y = 0$, $b^T y = -1$ so the primal is infeasible.
   (b) If $c^T\bar{x} < 0$, then $(x, s) = -\frac{1}{c^T\bar{x}}(\bar{x}, \bar{s}) \in \mathbf{R}^n \times \mathcal{K}$ satisfies $Ax + s = 0$, $c^T x = -1$ so the dual is infeasible.
   (c) If both $b^T\bar{y} < 0$ and $c^T\bar{x} < 0$, then both the primal and dual are infeasible.

3. If $\bar{\tau} = \bar{\kappa} = 0$ and one of the quantities $c^T\bar{x}$ or $b^T\bar{y}$ is negative, then a certificate of primal or dual infeasibility can be constructed as above. Otherwise nothing can be concluded about the original problem.

To avoid the inconclusiveness in the case when $\tau = \kappa = 0$, algorithms that solve the primal-dual pair (1) by finding a solution of the embedding (3) should be designed in a way such that they are not attracted to points with both κ and τ equal to zero.

Above we discussed how to extract information of the primal-dual pair (1) from a solution to the embedding (3). We can also go in the other direction. More precisely, if $(\bar{x}, \bar{y}, \bar{s})$ satisfies the KKT-conditions (2), then

$$(x, y, s, \tau, \kappa) = (\bar{x}, \bar{y}, \bar{s}, 1, 0) \tag{4}$$

solves the embedding. If $\bar{y}$ is a certificate for primal infeasibility, then

$$(x, y, s, \tau, \kappa) = (0, \bar{y}, 0, 0, 1) \tag{5}$$

solves the embedding. Finally, if $(\bar{x}, \bar{s})$ is a certificate for dual infeasibility, then

$$(x, y, s, \tau, \kappa) = (\bar{x}, 0, \bar{s}, 0, 1) \tag{6}$$

solves the embedding.

2.2 Normalized residual map

The essential idea of the refinement algorithm is to express the embedding (3) as the problem of finding a zero of a certain operator. Our goal now is to derive the form of this operator. To simplify notation, denote the skew-symmetric matrix in (3) by Q. Also define the cone $\boldsymbol{\mathcal{K}} = \mathbf{R}^n \times \mathcal{K}^* \times \mathbf{R}_+$ and its dual cone $\boldsymbol{\mathcal{K}}^* = \{0\}^n \times \mathcal{K} \times \mathbf{R}_+$. The embedding (3) with the additional constraint that $\tau + \kappa > 0$ can then be expressed as finding $u, v \in \mathbf{R}^{n+m+1}$ satisfying

$$Qu = v,\ u \in \boldsymbol{\mathcal{K}},\ v \in \boldsymbol{\mathcal{K}}^*,\ u_{m+n+1} + v_{m+n+1} > 0. \tag{7}$$

Finding u and v satisfying $Qu = v$ and $u_{m+n+1} + v_{m+n+1} > 0$ is easy.[1] The difficulty arises due to the constraints $u \in \boldsymbol{\mathcal{K}}$, $v \in \boldsymbol{\mathcal{K}}^*$. These constraints will be eliminated by using a parameterization. To describe the parameterization, consider the so-called *conic complementarity set* $\mathcal{C} \subseteq \mathbf{R}^{n+m+1} \times \mathbf{R}^{n+m+1}$ defined as

$$\mathcal{C} = \{(u, v) \in \boldsymbol{\mathcal{K}} \times \boldsymbol{\mathcal{K}}^* \mid u^T v = 0\}.$$

If u and v satisfy $Qu = v$, then $u^T v = u^T Q^T u = 0$, where we in the last equality have used the skew-symmetry of Q. Hence, the embedding (7) implies the constraints

$$Qu = v,\ (u, v) \in \mathcal{C},\ u_{m+n+1} + v_{m+n+1} > 0. \tag{8}$$

[1]For instance, u and v can be found as follows. Let (x, y, s) satisfy $A^T y = 0$, $Ax + s = 0$. If $b^T y = 0$ and $c^T x < 0$, let $u = (x, y, 0)$, $v = (0, s, -c^T x)$. If $b^T y < 0$ and $c^T x < 0$, let $u = (x, (c^T x / b^T y) y, 0)$ and $v = (0, s, -2c^T x)$. If $b^T y = 0$ and $c^T x = 0$, let $u = (x, y, 1)$ and $v = (c, s + b, 0)$. If $b^T y < 0$ and $c^T x = 0$, let $u = (x, y, 0)$ and $v = (0, s, -b^T y)$.

Conversely, (8) clearly implies (7) so we conclude that they are equivalent.

Next we want to parametrize the constraint $(u, v) \in \mathcal{C}$. To this end, first define the operators $\Pi : \mathbf{R}^{m+n+1} \to \mathbf{R}^{m+n+1}$ resp. $\Pi^* : \mathbf{R}^{m+n+1} \to \mathbf{R}^{m+n+1}$ as the Euclidean projections onto $\mathcal{K}$ resp. $-\mathcal{K}^*$. Since $\mathcal{K}$ is a closed convex cone its Moreau decomposition is given by

$$\Pi + \Pi^* = I,$$

where $I : \mathbf{R}^{n+m+1} \to \mathbf{R}^{n+m+1}$ denotes the identity operator. Consider the mapping $M : \mathbf{R}^{m+n+1} \to \mathcal{C}$ given by

$$M(z) = (\Pi z, -\Pi^* z).$$

This mapping is referred to as *Minty's parameterization of $\mathcal{C}$*. An elementary proof of the statement that M parameterizes $\mathcal{C}$ is given in the appendix. Using this parameterization the problem of finding u and v satisfying (8) can be expressed as the problem of finding $z \in \mathbf{R}^{m+n+1}$ satisfying

$$Q\Pi z = -\Pi^* z, \ (\Pi z - \Pi^* z)_{n+m+1} > 0.$$

The last of these two constraints can be slightly simplified. By using the Moreau decomposition of $\mathcal{K}$ in the first equality below we get that

$$(\Pi z - \Pi^* z)_{m+n+1} = (2\Pi z - z)_{m+n+1} = 2\max\{z_{n+m+1}, 0\} - z_{m+n+1} = |z_{m+n+1}| \geq 0,$$

with equality if and only if $z_{m+n+1} = 0$. Hence, we conclude that the problem of finding u and v satisfying (8) is equivalent to the problem of finding z satisfying

$$Q\Pi z = -\Pi^* z, \ z_{m+n+1} \neq 0. \tag{9}$$

Finding z satisfying $Q\Pi z = -\Pi^* z$ can be interpreted as finding a zero of the operator $\mathcal{R} : \mathbf{R}^{m+n+1} \to \mathbf{R}^{m+n+1}$ given by

$$\mathcal{R}(z) = Q\Pi z + \Pi^* z = \Big((Q - I)\Pi + I\Big)z.$$

However, a root-finding algorithm for solving $\mathcal{R}(z) = 0$ may be attracted to points with $z_{m+n+1} = 0$. To avoid this behaviour we instead consider the operator $\mathcal{N} : \{z \in \mathbf{R}^{m+n+1} \mid z_{m+n+1} \neq 0\} \to \mathbf{R}^{n+m+1}$ given by

$$\mathcal{N}(z) = \frac{1}{|z_{m+n+1}|}\mathcal{R}(z). \tag{10}$$

We refer to the operator $\mathcal{N}$ as the *normalized residual map*. The normalization with $|z_{m+n+1}|$ ensures that an algorithm for solving $\mathcal{N}(z) = 0$ is not attracted to points with $z_{m+n+1} = 0$.

2.3 Refinement algorithm

In the last subsection we reduced the problem of solving the primal-dual pair (1) to the problem of finding a zero of the normalized residual map $\mathcal{N}$. The idea of the refinement algorithm is to take an approximate zero of $\mathcal{N}$ and apply a root-finding algorithm to compute a more accurate zero. Root-finding algorithms are often based on derivative information. However, the operators projecting onto *e.g.* the second-order or positive semidefinite cones are not differentiable everywhere. Nevertheless, at many points in the space the projection operators are differentiable, see [2] and the references therein. As a heuristic idea for refinement, the derivative information at these points can be used and points of non-differentiability can be handled separately.

Having the differentiability issues in mind, the refinement algorithm can be summarized by the following steps.

1. Solve the primal-dual pair (1) using SCS. SCS either returns an approximate solution or a certificate for primal or dual infeasibility.

2. Use equations (4)-(6) to compute (u, v) corresponding to the information that SCS returns. Then compute the corresponding z, denoted by z_{approx}, according to $z_{\text{approx}} = u - v$.[2]

3. Starting from $z = z_{\text{approx}}$ and the assumption that $\mathcal{N}$ is differentiable at z, use LSQR to compute a search direction $\delta \in \mathbf{R}^{m \times n+1}$ that approximately minimizes

$$\|\mathcal{N}(z) + D_z\mathcal{N}(\delta)\|_2^2 + \lambda\|\delta\|_2^2.$$

   Here $\lambda > 0$ is a regularization parameter and the notation $D_z\mathcal{N}(\delta)$ refers to the differential of $\mathcal{N}$ at z evaluated in the direction δ.

4. Given the search direction δ, use backtracking line search to compute a new z.

5. Denote the refined zero by z_{ref}. Compute $(u_{\text{ref}}, v_{\text{ref}}) = M(z_{\text{ref}})$ and return the corresponding $(x_{\text{ref}}, y_{\text{ref}}, s_{\text{ref}})$.

For details of how to evaluate the differential of $\mathcal{N}$ at z in the direction δ, see [2].

3 Software

We present a C++ implementation of the refinement algorithm, with an interface to Python and CVXPY. The package, referred to as `coneref`, supports any cone that can be expressed as the Cartesian product of the zero cone, positive orthant, second-order cone, positive semidefinite cone, exponential cone, and dual exponential cone. The package exposes the function

```
cvxpy_solve(prob, ref_iter, lsqr_iter, verbose_scs, scs_opts
        warmstart, verbose_ref1, verbose_ref2),
```

where `prob` is a problem in cvxpy-format, `ref_iter` is the number of refinement steps and `lsqr_iter` is the number of LSQR iterations used in each refinement step. The rest of the parameters control output to the terminal; see the GitHub repository for details. The default values correspond to two refinement steps and 300 LSQR iterations.

[2]See Appendix A why z corresponding to (u, v) is given by $z = u - v$.

4 Numerical results

The aim of this section is to present empirical results of the refinement algorithm on several conic linear programs arising in practice. For the experiments we use the default settings of SCS and the default settings of the refinement strategy.

We consider the following problem classes: lasso, portfolio optimization, experiment design, robust principal component analysis and sparse inverse covariance estimation. For the exact formulations of the problem classes, see Appendix B.

For each problem class we distinguish between smaller and larger problem instances. For details of how the problem instances are generated, see Appendix C.

The numerical experiments presented here only consider bounded and feasible problems. Hence, to evaluate the effect of the refinement algorithm it makes sense to use the KKT-residual. In the experiments below the label "$\|$KKT-residual$\|_2$, refined" represents the quantity

$$\left(\|Ax+s-b\|_2^2 + \left\|A^T y + c\right\|_2^2 + (c^T x + b^T x)^2\right)^{1/2},$$

where (x, y, s) is the refined solution.

4.1 Smaller problem instances

For the effect of refinement on the group of smaller problem instances, see Figure 1. In Table 1-5 we also present the primal and dual feasibility residuals as well as the optimality gap before and after the refinement algorithm. In Table 1-5 we in particular see that the refinement algorithm often reduces the optimality gap by orders of magnitude.

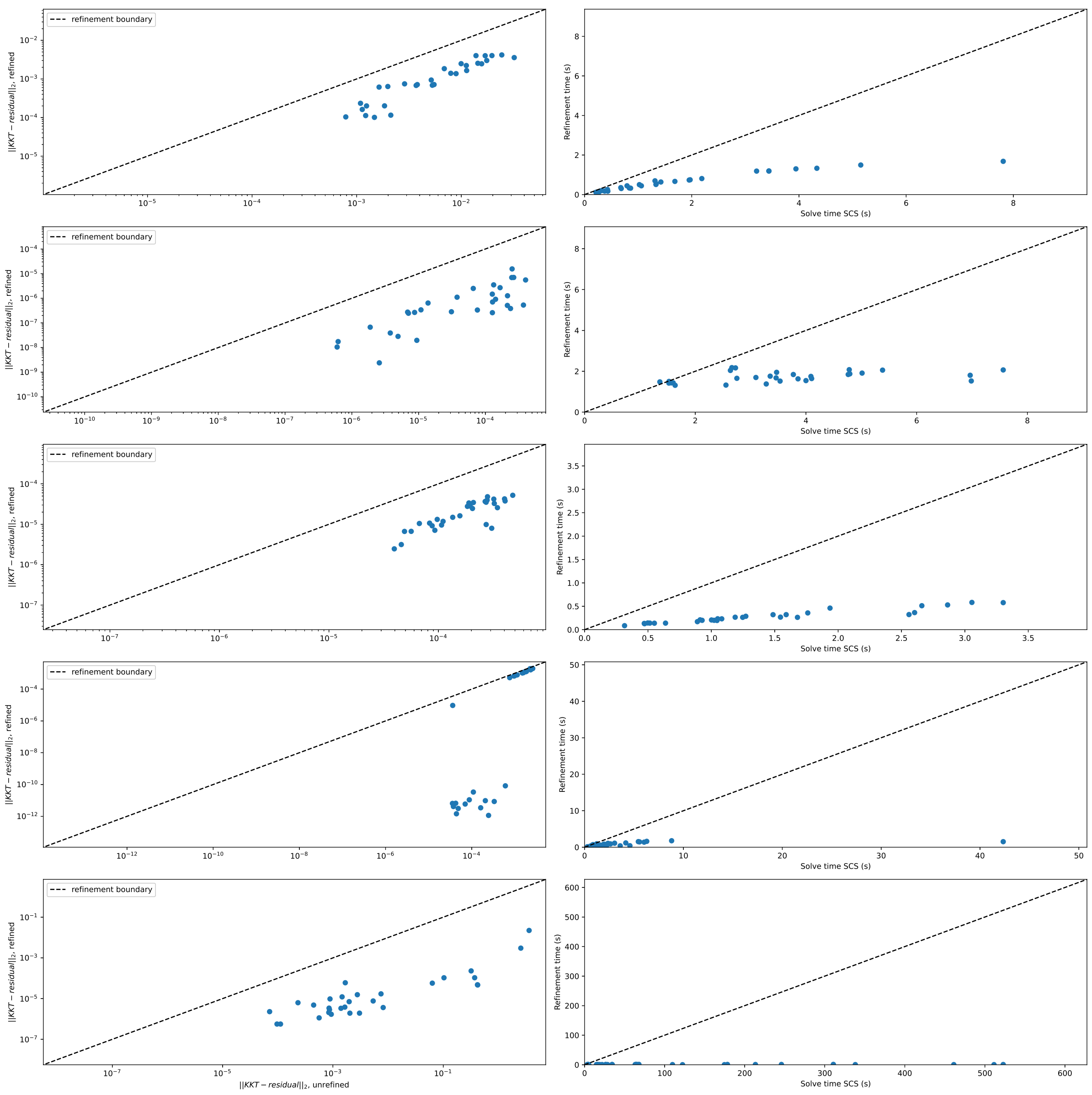

Figure 1: The effect of refinement on the group of smaller problem instances. Each row represents one problem class. The rows are ordered in the following order: lasso, portfolio optimization, experiment design, robust PCA and sparse inverse covariance estimation. Each blue dot represents one generated problem instance. The left figure in each row shows the effect of refinement on the norm of the KKT-residual, and the right figure in each row shows the solver time for SCS vs. the refinement time.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	1.556e-03	6.485e-04	2.598e-03	2.247e-04	-4.331e-03	-7.504e-08	7.701e+00
2	3.860e-03	3.565e-03	7.639e-04	2.289e-04	-3.181e-02	-3.910e-07	8.972e+00
3	2.457e-03	7.203e-04	6.460e-04	1.937e-04	1.334e-03	-8.753e-08	3.847e+00
4	2.875e-03	2.203e-03	3.383e-04	3.033e-04	-1.079e-02	-2.604e-05	5.021e+00

Table 1: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of lasso.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	6.868e-06	6.972e-07	2.852e-06	1.233e-07	1.279e-04	5.509e-09	1.810e+02
2	6.761e-06	2.732e-07	1.114e-06	2.901e-08	-4.344e-08	7.561e-10	2.495e+01
3	7.704e-07	6.668e-08	3.807e-07	8.291e-09	-1.679e-06	-1.982e-10	2.807e+01
4	1.200e-05	9.074e-07	3.968e-06	8.270e-08	1.416e-04	3.842e-09	1.560e+02

Table 2: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of portfolio optimization.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	4.761e-04	5.065e-05	3.264e-05	1.244e-05	-9.171e-07	-1.075e-08	9.151e+00
2	1.795e-04	2.559e-05	7.596e-05	1.201e-05	-4.516e-07	2.772e-08	6.895e+00
3	2.074e-04	2.492e-05	2.767e-04	6.433e-06	-4.345e-06	1.453e-07	1.344e+01
4	2.137e-04	7.593e-06	2.201e-04	2.423e-06	-2.072e-06	6.671e-08	3.849e+01

Table 3: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of experiment design.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	3.288e-04	4.161e-12	4.399e-05	7.508e-12	2.385e-05	5.684e-14	3.874e+07
2	1.841e-03	1.290e-03	1.491e-05	3.190e-06	2.488e-04	2.447e-08	1.440e+00
3	1.114e-03	7.840e-04	5.501e-06	1.947e-06	2.111e-04	1.182e-08	1.446e+00
4	1.542e-05	5.434e-06	7.120e-07	7.686e-06	3.267e-05	7.418e-12	3.838e+00

Table 4: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of robust PCA.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	2.158e-05	5.428e-05	6.338e-02	1.856e-05	-1.689e-03	2.410e-08	1.105e+03
2	2.125e-06	1.691e-06	4.148e-04	1.495e-07	-8.353e-04	2.718e-09	5.495e+02
3	7.593e-05	1.060e-04	2.569e-03	3.083e-07	-3.715e-01	9.938e-08	3.506e+03
4	6.147e-06	7.011e-06	1.718e-03	8.486e-07	9.635e-04	-5.375e-08	2.789e+02

Table 5: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of sparse inverse covariance estimation.

4.2 Larger problem instances

For the effect of refinement on the group of larger problem instances, see Figure 2 and Tables 6-10. It should be mentioned that SCS struggles with solving the problem of sparse inverse covariance estimation; for several problem instances SCS terminates with an inaccurate solution after reaching 10^5 iterations.

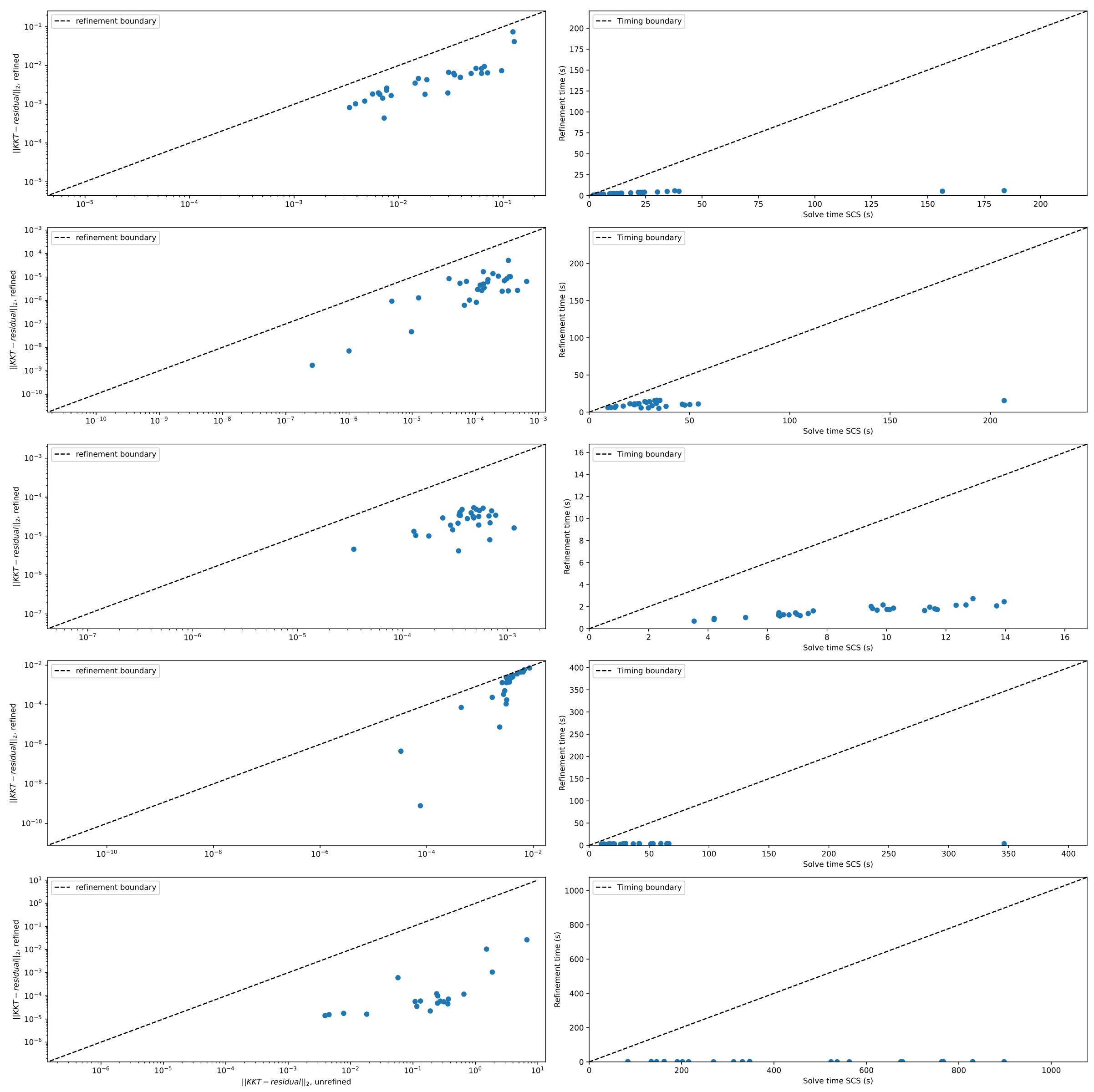

Figure 2: The effect of refinement on the group of larger problem instances. Each row represents one problem class. The rows are ordered in the following order: lasso, portfolio optimization, experiment design, robust PCA and sparse inverse covariance estimation. Each blue dot represents one generated problem instance. The left figure in each row shows the effect of refinement on the norm of the KKT-residual, and the right figure in each row shows the solver time for SCS vs. the refinement time.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	2.699e-03	2.689e-04	4.472e-03	3.481e-04	5.063e-03	-9.055e-07	1.654e+01
2	5.244e-03	2.182e-03	1.954e-03	7.026e-04	5.242e-03	1.196e-07	3.345e+00
3	5.569e-03	4.561e-03	1.442e-03	6.173e-04	1.436e-02	3.582e-06	3.361e+00
4	4.727e-03	1.239e-03	7.104e-03	1.311e-03	-1.573e-02	8.338e-07	9.921e+00

Table 6: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of lasso.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	4.247e-05	1.673e-05	1.312e-05	1.423e-06	-1.254e-04	-1.964e-08	7.923e+00
2	3.553e-05	6.469e-06	3.650e-05	2.544e-06	-2.834e-04	-7.072e-08	4.142e+01
3	1.954e-05	4.667e-06	3.997e-05	2.702e-06	-3.540e-05	-9.334e-09	1.054e+01
4	1.176e-05	2.671e-06	8.791e-06	3.044e-07	4.632e-04	7.670e-09	1.724e+02

Table 7: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of portfolio optimization.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	5.302e-04	1.604e-05	5.403e-06	1.055e-05	-7.189e-07	-1.712e-07	2.762e+01
2	4.663e-04	2.943e-05	5.580e-05	1.209e-05	-6.578e-07	1.594e-08	1.476e+01
3	2.829e-04	1.791e-05	4.032e-05	6.275e-06	-1.886e-08	3.813e-08	1.506e+01
4	2.817e-04	1.820e-05	1.842e-04	1.133e-05	-9.758e-08	-1.790e-08	1.570e+01

Table 8: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of experiment design.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	7.258e-05	1.351e-10	2.172e-05	7.671e-10	6.623e-06	5.684e-14	9.763e+04
2	2.048e-03	1.417e-03	1.406e-06	2.655e-05	2.921e-03	1.635e-09	2.517e+00
3	5.734e-03	4.284e-03	1.949e-06	1.516e-03	2.916e-03	7.291e-04	1.398e+00
4	6.217e-03	5.078e-03	5.643e-06	2.579e-03	2.597e-03	1.471e-03	1.145e+00

Table 9: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of robust PCA

.

	Primal residual		**Dual residual**		**Optimality gap**		
Problem instance	unrefined	refined	unrefined	refined	unrefined	refined	Empirical refinement factor
1	3.236e-06	1.534e-05	4.384e-03	9.796e-07	-1.102e-03	-4.580e-08	2.941e+02
2	1.211e-04	7.616e-05	6.548e-01	9.053e-05	-4.146e-03	-5.961e-07	5.535e+03
3	1.814e-03	4.871e-02	8.023e+01	7.238e-02	-4.184e+01	-7.675e-03	1.033e+03
4	9.917e-05	4.448e-05	3.619e-01	5.902e-06	-4.363e-03	1.593e-08	8.066e+03

Table 10: The residuals of the KKT-conditions for the unrefined vs. refined solution for 4 randomly chosen problem instances of sparse inverse covariance estimation.

Appendices

A Parameterization of conic complementarity set

Let $\mathcal{K} \subseteq \mathbf{R}^m$ be an arbitrary closed, convex cone (not necessarily the embedded cone denoted by $\boldsymbol{\mathcal{K}}$ in Section 2). Here we show that the conic complementarity set

$$\mathcal{C} = \{(u, v) \in \mathcal{K} \times \mathcal{K}^* \mid u^T v = 0\}$$

can be parameterized by the mapping $M : \mathbf{R}^m \to \mathcal{C}$ given by

$$M(z) = (\Pi z, -\Pi^* z),$$

where $\Pi : \mathbf{R}^m \to \mathbf{R}^m$ resp. $\Pi^* : \mathbf{R}^m \to \mathbf{R}^m$ are the Euclidean projections to $\mathcal{K}$ resp. $-\mathcal{K}^*$.

First note that M is well defined in the sense that $M(z) \in \mathcal{C}$ for any $z \in \mathbf{R}^m$. To show this we note that for any $z \in \mathbf{R}^m$,

$$(\Pi z)^T (\Pi^* z) = (z - \Pi^* z)^T \Pi^* z = 0,$$

where we in the first equality use the Moreau decomposition of $\mathcal{K}$ and in the second equality use properties of the Euclidean projection. What remains to prove is that for any $(u, v) \in \mathcal{C}$ there exists $z \in \mathbf{R}^m$ such that $M(z) = (u, v)$. To prove this we let $(u, v) \in \mathcal{C}$ be fixed and consider $\bar{z} := u - v$. Note that

$$\Pi(\bar{z}) = u \tag{11}$$

$$\Pi^*(\bar{z}) = -v. \tag{12}$$

Equations (11) and (12) follow from the definition of the dual cone. More precisely, to prove (11) note that for any $x \in \mathcal{K}$,

$$\begin{aligned} \|x - \bar{z}\|_2^2 &= \|x - u\|_2^2 + 2(x - u)^T v + \|v\|_2^2 \\ &= \|x - u\|_2^2 + 2x^T v + \|v\|_2^2 \geq \|v\|_2^2 \end{aligned}$$

with equality if and only if $x = u$. Hence,

$$\Pi(\bar{z}) = \operatorname*{argmin}_{x \in \mathcal{K}} \|x - \bar{z}\|_2^2 = u.$$

A similar argument proves equation (12).

B Problem classes

B.1 Lasso

Consider the optimization problem

$$\text{minimize } \frac{1}{2}\|Fz - g\|_2^2 + \mu\|z\|_1,$$

with variable $z \in \mathbf{R}^p$, and where $F \in \mathbf{R}^{q \times p}$, $g \in \mathbf{R}^q$ and $\mu \in \mathbf{R}_+$ are data.

B.2 Portfolio optimization

Consider a simple long-only portfolio optimization problem, in which we choose the relative weights of assets to maximize the expected risk-adjusted return of a portfolio. This amounts to solving the optimization problem [4]

$$\begin{array}{ll} \text{maximize} & \mu^T z - \gamma(z^T \Sigma z) \\ \text{subject to} & \mathbf{1}^T z = 1 \\ & z \geq 0, \end{array} \tag{13}$$

where the variable $z \in \mathbf{R}^p$ represents the portfolio of p assets, $\mu \in \mathbf{R}^p$ is the vector of expected return, $\gamma > 0$ is the risk aversion parameter, and $\Sigma \in \mathbf{R}^{p \times p}$ is the asset return covariance matrix (also known as the *risk model*). The risk model is expressed in factor model form

$$\Sigma = FF^T + D,$$

where $F \in \mathbf{R}^{p \times q}$ is the *factor loading matrix* and $D \in \mathbf{R}^{p \times p}$ is a diagonal matrix representing the asset-specific risk.

B.3 Experiment design

Consider a D-optimal design problem of the form

$$\begin{array}{ll} \text{minimize} & \log\det\Big(\sum_{i=1}^{p} \lambda_i H_i\Big)^{-1} \\ \text{subject to} & \mathbf{1}^T \lambda = 1 \\ & \lambda \geq 0, \end{array} \tag{14}$$

with variable $\lambda \in \mathbf{R}^p$ and problem data $H_i \in \mathbf{S}^d_+$, $i = 1, \ldots, p$.

B.4 Robust principal component analysis

We consider the optimization problem [4]

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\Big(\mathbf{Tr}(W_1) + \mathbf{Tr}(W_2)\Big) \\ \text{subject to} & -t \leq \mathbf{vec}(S) \leq t \\ & L + S = M \\ & \mathbf{1}^T t \leq \mu \\ & \begin{bmatrix} W_1 & L \\ L^T & W_2 \end{bmatrix} \succeq 0 \end{array}$$

with variables $L \in \mathbf{R}^{p \times q}$, $S \in \mathbf{R}^{p \times q}$, $W_1 \in \mathbf{R}^{p \times p}$, $W_2 \in \mathbf{R}^{q \times q}$, and $t \in \mathbf{R}^{pq}$, where $\mathbf{vec}(S)$ returns the columns of S stacked as a single vector. The matrix $M \in \mathbf{R}^{p \times q}$ is the problem data.

B.5 Sparse inverse covariance estimation

Suppose that $z_1, \dots, z_p$ are i.i.d $\mathcal{N}(0, \Sigma)$ with Σ^{-1} known to be sparse. In this setting the covariance matrix $\Sigma \in \mathbf{S}^q_+$ can be estimated by solving the optimization problem [6]

$$\text{minimize} \;\; -\log\det(S) + \mathbf{Tr}(SQ) + \alpha\|S\|_1,$$

where $S \in \mathbf{S}^q$ is the variable, $Q = \frac{1}{p}\sum_{\ell=1}^p z_\ell z_\ell^T$ is the sample covariance and $\alpha > 0$ is a hyperparameter. We then take $\hat{\Sigma}^{-1}$ as an estimate of Σ. Here $\|S\|_1$ denotes the elementwise ℓ_1-norm.

C Data generation

C.1 Lasso

We generated data for the numerical instances as follows. First, the entries of F were sampled independently from a standard normal distribution. We randomly generated a sparse vector $\hat{z}$ with p entries, only $p/10$ of which were nonzero. We then set $g = F\hat{z} + w$, where the entries in w were sampled independently and identically from $N(0, 0.1)$. We chose $\mu = 0.1\mu_{\max}$ for all instances, where $\mu_{\max} = \left\|F^T g\right\|_\infty$ is the smallest value of μ for which the solution to the problem is zero.

For the small problem instances, p is drawn uniformly from the interval 500-1500, and for the large problem instances p is drawn uniformly from the interval 1500-3000. For both the small and large problem instances we use $q = sp$ where s is drawn uniformly from 0.10-0.25.

C.2 Portfolio optimization

The vector of log-returns was sampled from a standard normal distribution, yielding log-normally distributed returns. The entries of F were sampled independently from $N(0, 0.1)$, and the diagonal entries of D were sampled independently from a uniform distribution on 0-0.1. For all problems we chose $\gamma = 1$.

For the small problem instances, p is drawn uniformly from the interval 2000-3000. For the larger problem instances, p is drawn uniformly from the interval 4000-7000. For both the small and large problem instances we let $q = \lceil p/10 \rceil$.

C.3 Experiment design

Each matrix H_i is chosen as $H_i = Q_i^T Q_i$ where the elements of $Q_i \in \mathbf{R}^{d\times d}$ are sampled i.i.d from the interval $[-1, 1]$.

For the small problem instances p resp. q are sampled uniformly from the intervals 500-1000 resp. 10-20. For the large problem instances p resp. q are sampled uniformly from the intervals 1000-2000 resp. 20-30.

C.4 Robust PCA

We set $M = \hat{L} + \hat{S}$ where $\hat{L}$ was a randomly generated rank-r matrix and $\hat{S}$ was a sparse matrix with approximately 10% nonzero entries. For all instances, we set μ to be equal to the sum of absolute values of the entries of $\hat{S}$ and generated the data with $r = 10$. For simplicity, we chose the matrices to be square, *i.e.,* , $p = q$ for all instances.

For the small problem instances, p is drawn uniformly from the interval 40-70 and $r = \lfloor p/10 \rfloor$. For the larger problem instances, p is drawn uniformly from the interval 80-100 and $r = 10$.

C.5 Sparse inverse covariance estimation

We generated $S \in \mathbf{S}^q_{++}$ with $q = 100$ and approximately 10% nonzero entries. Then we calculated Q using p i.i.d. samples from $N(0, S^{-1})$.

For the small problem instances, q is drawn uniformly from the interval $[50, 70]$. For the larger problem instances, q is drawn uniformly from the interval $[80, 110]$. For both the small and large problem instances we choose $p = sq$ where s is drawn uniformly from the interval $[7, 13]$. For each problem instance the matrix S had between 10-20% nonzero entries. The regularization parameter α was chosen as $\alpha = 0.005\alpha_{\max}$, where $\alpha_{\max} = \sup_{i \neq j} |Q_{ij}|$ is the smallest α for which the solution is trivially the diagonal matrix $(\mathrm{diag}(Q) + \alpha I)^{-1}$.

References

[1] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[2] E. Busseti, W. Moursi, and S. Boyd. Solution refinement at regular points of conic problems. *Computational Optimization and Applications*, 74(3):627–643, 2019.

[3] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[4] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016.

[5] Yinyu Ye, Michael J. Todd, and Shinji Mizuno. An o(n l)-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994.

[6] Junzi Zhang, Brendan O'Donoghue, and Stephen Boyd. Globally convergent type–I Anderson acceleration for non-smooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4):3170–3197, 2020.