# 1    Overview

This section discusses some of the algorithms that have recently been proposed in the literature to solve the Robust PCA problem

$$p^* = \min_{L,S} \|L\|_* + \lambda\|S\|_1$$
$$\text{s.t.} \quad M = L + S \tag{1}$$

There are various methods that can be used to solve (1). A straightforward way described in section 2.1 is to use general purpose interior point solvers [17, 18] to solve an SDP formulation of the dual of (1), an approach that works well for low-dimensional data $M$ but unfortunately does not scale well. Another approach is to use iterative thresholding techniques as described in section 2.2, which result in a very simple algorithm that can be applied also to high-dimensional data. Unfortunately the convergence of this algorithm is extremely slow. More sophisticated approaches include an accelerated proximal gradient algorithm (section 2.3) and a gradient ascent algorithm applied to the dual problem of (1) (section 2.4). The current state of the art seems to be a adaptation of the Augmented Lagrangian Method to the non-smooth problem (1), an approach that is discussed in section 2.5.

# 2    Main algorithms for Robust PCA

## 2.1    Interior Point Methods

As will be shown in the following section, the dual problem of (1) can be cast as a Semidefinite Programming (SDP) problem for which a number of efficient polynomial-time interior-point solvers have been developed. In principle, one can then just apply these general purpose off-the-shelf solvers to the dual SDP. This approach will work well for small and medium sized problems, but unfortunately it does not scale with the size of the data matrix $M$. The following section first develops the dual of the Robust PCA problem, section 2.1.2 then discusses the limitations of using interior point methods on the resulting SDP.

### 2.1.1    Formulation of the Dual problem of Robust PCA as an SDP

Let us use the equality constraint to eliminate the variable $L$ from (1). Using the facts that $\|X\|_* = \max_{\|Y\|_2 \leq 1} \mathbf{Tr}\, Y^T X$ and $\|X\|_1 = \max_{\|Z\|_\infty \leq 1} \mathbf{Tr}\, Z^T X$ the problem can be written as

$$p^* = \min_{S} \max_{Y,Z} \quad \mathbf{Tr}\, Y^T(M - S) + \lambda\,\mathbf{Tr}\, Z^T S$$
$$\text{s.t.} \quad \|Y\|_2 \leq 1 \tag{2}$$
$$\|Z\|_\infty \leq 1$$

The dual function is

$$g(Y, Z) = \min_{S} \quad \mathbf{Tr}\, Y^T M + \mathbf{Tr}(\lambda Z - Y)^T S$$
$$= \begin{cases} \mathbf{Tr}\, Y^T M & \text{if } \lambda Z = Y \\ -\infty & \text{otherwise} \end{cases} \tag{3}$$

We obtain the dual problem

$$p^* \geq d^* = \max_{Y,Z} \quad \mathbf{Tr}\, Y^T M$$
$$\text{s.t.} \quad \lambda Z = Y$$
$$\|Y\|_2 \leq 1$$
$$\|Z\|_\infty \leq 1$$

(4)

which after eliminating the variable $Z$ and using homogeneity of the norm becomes

$$d^* = \max_{Y} \quad \mathbf{Tr}\, Y^T M$$
$$\text{s.t.} \quad \|Y\|_2 \leq 1$$
$$\|Y\|_\infty \leq \lambda$$

(5)

Noting that $\|Y\|_2 \leq 1 \iff I - X^T(I)^{-1}X \succeq 0$ and using a Schur Complement lemma, the dual problem can be transformed into the following SDP standard form:

$$d^* = \max_{Y} \quad \mathbf{Tr}\, M^T Y$$
$$\text{s.t.} \quad \begin{bmatrix} I & Y^T \\ Y & I \end{bmatrix} \succeq 0$$
$$\mathbf{Tr}\, \Delta_{ij}^T Y \leq \lambda, \quad i = 1, \ldots, m,\ j = 1, \ldots, n$$
$$\mathbf{Tr}\, \Delta_{ij}^T Y \geq -\lambda, \quad i = 1, \ldots, m,\ j = 1, \ldots, n$$

(6)

Here $\Delta_{ij} \in \mathbf{R}^{m \times n}$ is such that $\Delta_{ij}(k, l) = \delta_{ik}\delta_{jl}$. Note that both primal and dual problem are (trivially) strictly feasible. In fact, the primal is unconstrained and an obvious strictly feasible dual variable is $Y = 0$. Hence strong duality holds ($p^* = d^*$) and both primal and dual problem are attained.

**Recovering the primal solution**

To recover the primal solution we can distinguish the two cases $\lambda \leq 1$ and $\lambda > 1$. For the latter one, first note that $\|Y\|_\infty \leq \|Y\|_2$ for all $Y$. To show this first recall that $c := \|Y\|_2 = \max_{\|x\|_2=1} \|Yx\|_2$. This means that $c - x^T Y^T Y x \geq 0$, $\forall x$ or, equivalently, $cI - Y^T Y \succeq 0$. Using a Schur Complement this can be written as

$$\begin{bmatrix} I & Y \\ Y^T & cI \end{bmatrix} \succeq 0$$

It is easy to see that this implies that

$$\begin{bmatrix} 1 & Y_{ij} \\ Y_{ij} & c \end{bmatrix} \succeq 0$$

for all $i, j$, which is equivalent to $|Y_{ij}| \leq c$, $\forall i, j$. This shows that $\|Y\|_\infty \leq \|Y\|_2$.

Hence we observe that if $\lambda > 1$ the constraint $\|Y\|_2 \leq \lambda$ will be inactive, that is $|Y_{ij}| < \lambda$, $\forall i, j$. Therefore, for $\lambda > 1$, removing this constraint from (5) does not change the problem. We have $d^* = \max_{\|Y\|_2 \leq 1} \mathbf{Tr}\, Y^T M = \|M\|_*$, and this lower bound is achieved by the primal (feasible) solution $L = M$ and $S = 0$.

If $\lambda \leq 1$, then ....

### 2.1.2 Using Interior Point Methods to Solve the Dual

Problem (5) is an SDP and can therefore in principle be solved using off-the-shelf solver packages like [17] and [18]. These solvers are based on interior point methods which offer superior convergence rates [6] but unfortunately do not scale particularly well with the size of the matrix $M$. The reason for this is that the computation of the step direction relies on second-order information of the (barrier-augmented) objective function, the computation of which is infeasible when the number of variables is very high. For applications that involve large-scale data matrices (with $m$ in the thousands or even millions), the use interior point methods therefore quickly becomes infeasible. In order to overcome this scalability issue, a variety of first-order methods exploiting the particular structure of the Robust PCA problem have been proposed. Some of these methods will be described in the following.

## 2.2 Iterative Thresholding Method

Among the first techniques used to solve (1) for high-dimensional data was an adaptation [19][1] of an iterative thresholding method originally proposed for the matrix completion problem in [7]. For this method the authors of consider a relaxation of the original problem (1) of the form

$$\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L\|_F^2 + \frac{1}{2\tau}\|S\|_F^2$$
$$\text{s.t.} \quad M = L + S \tag{7}$$

where $\tau \gg 1$ is a scalar parameter. Generalizing the result from [7] (which considers the matrix completion problem, i.e. $S = 0$), the authors of [19] argue that for large values of $\tau$ the solution of (7) will be very close to that of (1) (but interestingly enough they do not prove it!). One can now use iterative thresholding techniques to solve (7). Although the resulting algorithm has little relevance in practice because of its extremely slow convergence, we will see in the following sections that its main ideas are the basis for a number of other similar algorithms. Therefore we briefly describe the iterative thresholding algorithm here.

The Lagrangian of the problem (7) is given by

$$\mathcal{L}(L, S, Y) = \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L\|_F^2 + \frac{1}{2\tau}\|S\|_F^2 + \frac{1}{\tau}\langle Y, M - L - S\rangle \tag{8}$$

The idea of the iterative thresholding algorithm is, as the name suggests, to update the variables $L, S$ and $Y$ iteratively. More specifically, the Lagrangian $\mathcal{L}(L, S, Y)$ is minimized w.r.t $L$ and $S$ for some fixed dual variable $Y$, and the violation of the constraint is then used to update $Y$ using the gradient step $Y^+ = Y + t(M - L - S)$, where $0 < t < 1$ is the step size.

---

[1]note that the iterative thresholding algorithm seems to never have appeared in the published version of [19], which instead contains the accelerated proximal gradient method described in section 2.3

Note that for fixed $Y$ we have

$$
\begin{aligned}
(\hat{L}, \hat{S}) :&= \arg\min_{L,S} \ \mathcal{L}(L, S, Y) \\
&= \arg\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L\|_F^2 + \frac{1}{2\tau}\|S\|_F^2 + \frac{1}{\tau}\langle Y, M - L - S\rangle \\
&= \arg\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \frac{1}{2\tau}\|L - Y\|_F^2 + \frac{1}{2\tau}\|S - Y\|_F^2
\end{aligned}
\tag{9}
$$

We note that the above problem is completely separable hence the minimizers $\hat{L}$ and $\hat{S}$ can be determined independently. Using optimality conditions based on the subgradient it can be shown [7, 19] that the minimizers $\hat{L}$ and $\hat{S}$ are both of a simple form. To this end, consider the following extension to the matrix case of the well known soft-thresholding operator (e.g. from the proximal mapping of the $l_1$-norm in the vector case):

$$
(\mathcal{S}_\varepsilon[X])_{ij} := \begin{cases} x_{ij} - \varepsilon & \text{if } x_{ij} > \varepsilon \\ x_{ij} + \varepsilon & \text{if } x_{ij} < -\varepsilon \\ 0 & \text{otherwise} \end{cases}
\tag{10}
$$

**Theorem 1** (Singular Value Thresholding [7]). *For each $\tau \geq 0$ and $Y \in \mathbf{R}^{m\times n}$, we have that*

$$
\mathcal{D}_\tau(Y) := U\mathcal{S}_\tau[\Sigma]V^T = \arg\min_{X} \ \|X\|_* + \frac{1}{2\tau}\|X - Y\|_F^2
\tag{11}
$$

*where $Y = U\Sigma V^*$ is the SVD of $Y$.*

Since Theorem 1 is central to all algorithms discussed in this chapter, we give its proof here.

*Proof of Theorem 1.* Since the objective function $f_0(X) := \arg\min_X \ \|X\|_* + \frac{1}{2\tau}\|X - Y\|_F^2$ is strictly convex its minimizer is unique. Recall that $Z$ is a subgradient of a convex function $f : \mathbf{R}^{m\times n} \mapsto \mathbf{R}$ at $X_0$ if $f(X) \geq f(X_0) + \langle Z, X - X_0\rangle$ for all $X$. The set of all sugradients $Z$ at $X_0$ is called the subdifferential and denoted $\partial f(X_0)$. By definition it follows that $\hat{X}$ minimizes $f_0$ if and only if $0$ is as subgradient of $f_0$ at $\hat{X}$. Using standard subgradient calculus this translates to $0 \in \hat{X} - Y + \tau\partial\|\hat{X}\|_*$, where $\partial\|\hat{X}\|_*$ is the subdifferential of the nuclear norm at $\hat{X}$. To show that $\mathcal{D}_\tau(Y)$ satisfies (11) we therefore need to show that $0 \in \mathcal{D}_\tau(Y) - Y + \tau\partial\|\mathcal{D}_\tau(Y)\|_*$.

Let $X \in \mathbf{R}^{m\times n}$ be arbitrary Let $U\Sigma V^*$ be its SVD. It can be shown that

$$
\partial\|X\|_* = \left\{ UV^* + W \mid W \in \mathbf{R}^{m\times n},\ U^*W = 0,\ WV = 0,\ \|W\|_2 \leq 1 \right\}
$$

To use the above result decompose the SVD of $Y$ as $Y = U_0\Sigma_0 V_0^* + U_1\Sigma_1 V_1^*$, where $U_0, V_0$ $(U_1, V_1)$ are the singular vectors associated with singular values greater (less or equal) than $\tau$. With this notation we have $\mathcal{D}_\tau(Y) = U_0(\Sigma_0 - \tau I)V_0^*$, so $Y - \mathcal{D}_\tau(Y) = \tau(U_0 V_0^* + W)$ with $W = \tau^{-1}U_1\Sigma_1 V_1^*$. By definition of the SVD $U_0^*W = 0$ and $WV_0 = 0$. Furthermore, since $\max(\Sigma_1) \leq \tau$ it also holds that $\|W\|_2 \leq 1$. Together this shows that $Y - \mathcal{D}_\tau(Y) \in \tau\partial\|\mathcal{D}_\tau(Y)\|_*$. $\qquad\square$

It is also not hard to find the minimizer $\hat{S}$ of (8), as the following proposition shows:
**Proposition 1** (Matrix Value Thresholding). *For each $\tau, \lambda \geq 0$ and $Y \in \mathbf{R}^{m\times n}$, we have that*

$$
\mathcal{S}_{\tau\lambda}[Y] = \arg\min_{X} \ \lambda\|X\|_1 + \frac{1}{2\tau}\|X - Y\|_F^2
\tag{12}
$$

*Proof of Proposition 1.* Note that the objective function $h_0(X) := \lambda\|X\|_1 + \frac{1}{2\tau}\|X - Y\|_F^2$ in (12) is completely decomposable and can be written in the form $h_0(X) = \sum_{i,j} \lambda|X_{ij}| + \frac{1}{2\tau}(X_{ij} - Y_{ij})^2$, so the minimization over $X$ can be carried out element-wise. Using the same reasoning as in the proof of Theorem 1, we have that the element $X_{ij}$ is a minimizer if and only if $0 \in \partial h_0(X)$. This condition can be written as $Y_{ij} - X_{ij} \in \tau\lambda\,\partial|X_{ij}|$. We consider the cases $X_{ij} = 0$ and $X_{ij} \neq 0$. If $X_{ij} = 0$ then the condition reads $Y_{ij} \in \tau\gamma[-1, 1]$. In other words, when $|Y_{ij}| \leq \tau\gamma$ then $X_{ij} = 0$. On the other hand, if $X_{ij} \neq 0$, then $\partial\,|X_{ij}| = \mathrm{sign}(X_{ij})$ and we have $X_{ij} = Y_{ij} - \tau\lambda\,\mathrm{sign}(Y_{ij})$, which is in fact the soft-thresholding operator $\mathcal{S}_{\tau\lambda}[X_{ij}]$. $\square$

Applying Theorem 1 and Proposition 1 to (9) we find that the minimizers $\hat{L}$ and $\hat{S}$ of (8) are given, respectively, by

$$\hat{L} = U\mathcal{S}_\tau[\Sigma]V^T \tag{13}$$

$$\hat{S} = \mathcal{S}_{\tau\lambda}[Y] \tag{14}$$

The overall iterative thresholding algorithm for Robust PCA is given in Algorithm 1.

---

**Algorithm 1:** Iterative Thresholding Algorithm

    **Input**: Observation matrix $M$, parameters $\lambda, \tau$
    initialization: $k = 0$, $Y_0 = 0$;
    **while** *not converged* **do**
        $k = k + 1$;
        $(U, \Sigma, V) = \mathrm{svd}(Y_{k-1})$;
        $L_k = U\mathcal{S}_\tau[\Sigma]V^T$;
        $S_k = \mathcal{S}_{\lambda\tau}[Y_{k-1}]$;
        $Y_k = Y_{k-1} + t_k(M - L_k - S_k)$;
    **Output**: $L = L_k$, $S = S_k$

---

Algorithm 1 is extremely simple to implement, as each iteration only requires the computation of an SVD of the current dual variable $Y_k$ and a few elementary matrix operations. Note that, in fact, since the shrinking operator sets all singular values less than the threshold parameter to zero one really only needs to compute the singular values that lie above this threshold. However, while this iterative thresholding scheme is very simple and has been proved to converge, its convergence unfortunately is extremely slow. The authors of [19] found that it typically requires about $10^4$ iterations to converge for problems of reasonable size. Furthermore it is hard to find good general schemes to optimize the choice of the step size $t_k$ [11]. Therefore the practical applicability of this approach is limited.

## 2.3 Accelerated Proximal Gradient Method

An accelerated proximal gradient (APG) method for solving (1) was proposed in [12]. The method is essentially an application of the FISTA algorithm [2] to a relaxation of the original RPCA problem in combination with a continuation technique. This FISTA algorithm algorithm is reminiscent of Nesterov's "optimal" first-oder method for smooth objective functions [14], whose $O(1/k^2)$ convergence rate result was extended to non-smooth objectives in [15]. Besides being theoretically

appealing, the APG method is also in practice much faster than the iterative thresholding algorithm discussed in section 2.2.

The following sections describe the APG algorithm. As in [12] we first give a general formulation and then show how it can be applied to the Robust PCA problem, using ideas from section 2.2.

### 2.3.1 A General Form of the Accelerated Proximal Gradient Method

Proximal gradient algorithms in general can be used to solve unconstrained problems of the form

$$\min_x \; f(x) := g(x) + h(x) \tag{15}$$

where $g$ is convex and differentiable and $h$ is closed[2], convex but need not be differentiable. The proximal mapping of a convex function $h$ is defined as

$$\text{prox}_h(x) := \arg\min_u \left( h(u) + \frac{1}{2} \|u - x\|^2 \right) \tag{16}$$

Here $\|\cdot\|$ denotes the inner product norm[3]. Note that since $h$ is convex and the norm is strictly convex, the optimizer in (16) is unique. The step of each iteration of the classic (non-accelerated) proximal gradient algorithm is

$$x^+ = \text{prox}_{\delta h}(x - \delta \nabla g(x)) \tag{17}$$

where $x^+$ denotes the next iterate, based on the current iterate $x$, and $\delta$ is the step size (which can be fixed or determined via backtracking line search). Since the proximal mapping (17) needs to be computed at each iteration (possibly multiple times because of the line search), the algorithm is most effective when this can be done cheaply. Depending on the function $h$, the proximal gradient algorithm can be seen as a generalization of different gradient-based algorithms. In particular, for $h(x) \equiv 0$ the standard gradient algorithm is recovered and for $h(x) = I_C(x)$, where $I_C$ denotes the indicator function for the convex set $C$, the projected gradient algorithm is recovered.

Note that (17) can be written as

$$
\begin{aligned}
x^+ &= \arg\min_u \left( h(u) + \frac{1}{2\delta} \|u - x + \delta \nabla g(x)\|^2 \right) = \arg\min_u \left( h(u) + \frac{1}{2\delta} \|u - G(x)\|^2 \right) \\
&= \arg\min_u \left( h(u) + g(x) + \langle \nabla g(x), u - x \rangle + \frac{1}{2\delta} \|u - x\|^2 \right)
\end{aligned}
\tag{18}
$$

where $G(x) := x - \delta \nabla g(x)$. Therefore, each step (17) can be interpreted as minimizing the function $h(u)$ plus a quadratic local model of $g(u)$ around the current iterate $x$. This classic form of the proximal gradient algorithm is well known in the literature and, under Lipschitz continuity of the gradient of $g$ and some technical conditions on the step size $t$, has been shown to have a convergence rate no worse than $O(1/k)$ (PUT REFERENCE). In particular, a popular choice for $\delta$ is the fixed step size $\delta = 1/L_g$, where $L_g$ is a Lipschitz constant of the gradient of $g$, that is we

---

[2]that is, its epigraph is closed

[3]Note that the derivation holds for real inner product spaces, not just $\mathbf{R}^n$. We are particularly interested in the case $\mathbf{R}^{m \times n}$

have $\|\nabla g(x) - \nabla g(y)\| \leq L_g \|x - y\|$ for all $x, y$. Note that in practice for a general problem the value of $L_g$ might be unknown. This is not a big concern for the type of problem arising in Robust PCA, hence we will assume this choice of $\delta = 1/L_g$ in the following.

It turns out that the choice for the point around which the quadratic local model of $g(u)$ is constructed has an important effect on the convergence rate, and that less obvious choices than just the previous iterate $x$ are actually better (in the sense that they yield higher convergence rates). Specifically, consider a generalized version of (18), where the approximation of $g$ is constructed around some $y$ that may depend on all prior iterates $x_0, \ldots, x_k$:

$$x^+ = \arg\min_u \left( h(u) + \frac{L_g}{2} \|u - G(y)\|^2 \right) \tag{19}$$

Nesterov in [14] showed that in the smooth case the standard gradient algorithm (i.e. $h(x) \equiv 0$) can be accelerated by choosing $y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$, leading to a theoretical convergence rate of $O(1/k^2)$. This algorithm is optimal in the sense that its convergence rate achieves (in terms of order) the theoretical complexity bound on all first order algorithms. The seminal work [14] has been extended to the non-smooth case in [2, 15], yielding the generic accelerated proximal gradient method given in Algorithm 2.

---

**Algorithm 2:** Accelerated Proximal Gradient Algorithm

initialization: $k = 0$, $t_0 = t_{-1} = 1$;
**while** *not converged* **do**
$\quad y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$;
$\quad G_k = y_k - \frac{1}{L_g} \nabla g(y_k)$;
$\quad x_{k+1} = \arg\min_x \left( h(x) + \frac{L_g}{2} \|x - G(y_k)\|^2 \right)$;
$\quad t_{k+1} = \frac{1 + \sqrt{4t_k^2 + 1}}{2}$;
$\quad k = k + 1$;

---

### 2.3.2 Accelerated Proximal Gradient Algorithm Applied to Robust PCA

Using some the ideas from the iterative thresholding method in section 2.2, it is possible to construct an accelerated proximal gradient algorithm for a relaxation of the original problem (1). In particular, it is easy to see that in the common case $h(x) = \|x\|_1$ the update for $x$ in Algorithm 2 is simply given by $x_{k+1} = \mathcal{S}_{1/L_g}[G(y_k)]$, where $\mathcal{S}_\varepsilon$ is the soft-thresholding operator defined in (10). Similar ideas can be used to develop an APG algorithm for the Robust PCA problem.

The authors of [12] consider a relaxation of (1) of the form

$$\min_{L,S} \ \mu\|L\|_* + \mu\lambda\|S\|_1 + \frac{1}{2}\|L + S - M\|_F^2 \tag{20}$$

It can be shown that for $\mu \to 0$, any solution of (20) approaches the solution set of (1). In practice, rather than fixing $\mu$ to some small value, one can achieve superior convergence of the algorithm by using a continuation technique on $\mu$, that is, by solving (20) by repeatedly decreasing the value of $\mu$ in the steps of the accelerated proximal gradient algorithm. This can be interpreted as a particular homotopy method.

---

With $X = (L, S)$ we identify $h(X) = \mu\|L\|_* + \mu\lambda\|S\|_1$ and $g(X) = \frac{1}{2}\|L + S - M\|_F^2$ in (20). The gradient of $g$ is $\nabla g(X) = (\nabla_L g(X), \nabla_S g(X))$, where $\nabla_L g(X) = \nabla_S g(X) = L + S - M$. Thus

$$\|\nabla g(X_1) - \nabla g(X_2)\| = \left\|\begin{bmatrix} L_1 + S_1 - M \\ L_1 + S_1 - M \end{bmatrix} - \begin{bmatrix} L_2 + S_2 - M \\ L_2 + S_2 - M \end{bmatrix}\right\| = \left\|\begin{bmatrix} L_1 - L_2 + S_1 - S_2 \\ L_1 - L_2 + S_1 - S_2 \end{bmatrix}\right\|$$

$$\leq \left\|\begin{bmatrix} L_1 - L_2 \\ S_1 - S_2 \end{bmatrix}\right\| + \left\|\begin{bmatrix} S_1 - S_2 \\ L_1 - L_2 \end{bmatrix}\right\| = 2\|X_1 - X_2\|$$

which means that a Lipschitz constant is given by $L_g = 2$. It turns out that because of the separability of both the function $h$ and the Frobenius norm the update can be be decomposed as follows:

$$(L_{k+1}, S_{k+1}) = X_{k+1} = \arg\min_X \left( h(X) + \frac{L_g}{2}\|X - G(Y_k)\|^2 \right)$$

$$= \arg\min_{L,S} \left( \mu\|L\|_* + \mu\lambda\|S\|_1 + \left\|\begin{bmatrix} L \\ S \end{bmatrix} - \begin{bmatrix} G_k^L \\ G_k^S \end{bmatrix}\right\|^2 \right)$$

$$= \arg\min_{L,S} \left( \|L\|_* + \frac{1}{2\frac{\mu}{2}}\left\|L - G_k^L\right\|^2 + \lambda\|S\|_1 + \frac{1}{2\frac{\mu}{2}}\left\|S - G_k^S\right\|^2 \right)$$

The problem is therefore completely separable and clearly the sum of two problems of the form (15). In particular, if $G_k^L = U\Sigma V^T$ is the SVD of $G_k^L$ then, using the results from section 2.2, the iterates $L_{k+1}$ and $S_{k+1}$ are given by

$$L_{k+1} = U\mathcal{S}_{\frac{\mu}{2}}[\Sigma]V^T \tag{21}$$

$$S_{k+1} = \mathcal{S}_{\frac{\lambda\mu}{2}}[G_k^S] \tag{22}$$

Therefore it is rather straightforward to formulate a version Algorithm 2 for the Robust PCA problem. One extension to Algorithm 2 is the use of a continuation technique (or homotopy method) for the parameter $\mu$. In [12], the authors propose to start from some large value $\mu_0$ and then choose $\mu_{k+1} = \max(\eta\mu_k, \bar{\mu})$, where $0 < \eta < 1$ and $\bar{\mu}$ is a lower limit on $\mu_k$. This continuation technique has been observed to improve convergence. The Accelerated Proximal Gradient Algorithm applied to the Robust PCA problem is given in Algorithm 3.

---

**Algorithm 3:** Accelerated Proximal Gradient Algorithm for Robust PCA

**Input**: Observation matrix $M$, parameter $\lambda$

initialization: $k = 0$, $L_0 = L_{-1} = 0$, $S_0 = S_{-1} = 0$, $t_0 = t_{-1} = 1$, $\bar{\mu} = \delta\mu_0$;

**while** *not converged* **do**

$\quad Y_k^L = L_k + \frac{t_{k-1}-1}{t_k}(L_k - L_{k-1})$, $\quad Y_k^S = S_k + \frac{t_{k-1}-1}{t_k}(S_k - S_{k-1})$;

$\quad G_k^L = Y_k^L - \frac{1}{2}(Y_k^L + Y_k^S - M)$;

$\quad (U, \Sigma, V) = \text{svd}(G_k^L)$, $L_{k+1} = U\mathcal{S}_{\mu_k/2}[\Sigma]V^T$;

$\quad G_k^S = Y_k^S - \frac{1}{2}(Y_k^L + Y_k^S - M)$;

$\quad S_{k+1} = \mathcal{S}_{\lambda\mu_k/2}[G_k^S]$;

$\quad t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$;

$\quad \mu_{k+1} = \max(\eta\mu_k, \bar{\mu})$;

$\quad k = k + 1$;

**Output**: $L = L_k$, $S = S_k$

---

We will restate the main convergence theorem for Algorithm 3 without proof.

**Theorem 2** ([11]). *Let* $F(X) = F(L, S) := \bar{\mu}\|L\|_* + \bar{\mu}\lambda\|S\|_1 + \frac{1}{2}\|L + S - M\|_F^2$. *Then for all* $k > k_0 := -C_1 \log(\eta)$ *it holds that*

$$F(X_k) - F(X^*) \leq \frac{4\|C_{k_0} - X^*\|_F^2}{(k - k_0 + 1)^2}$$

*where* $C_1 = \log(\mu_0/\bar{\mu})$ *and* $X^*$ *is any solution to* (20).

*Remark* 2.1 (Accuracy of the APG method). Note that for any finite $\bar{\mu}$ Algorithm 3 provides only an approximate solution of the Robust PCA problem due to the replacement of the equality constraint with the penalty term $\frac{1}{2\bar{\mu}}\|L + S - M\|_F^2$.

## 2.4 Gradient Ascent on the Dual

In section 2.1.2 we discussed why solving the Robust PCA dual problem (5) via interior point techniques quickly becomes infeasible with growing size of the data matrix $M$. Another way of solving the dual given in [12] is based on a steepest ascent algorithm. Note that the dual problem (5) derived in section 2.1.1 can be written as

$$d^* = \max_Y \ \mathbf{Tr}\, M^T Y \quad \text{subject to} \quad J(Y) \leq 1 \tag{23}$$

where $J(Y) := \max\left(\|Y\|_2, \lambda^{-1}\|Y\|_\infty\right)$. Denote by $F := \{Y \mid J(Y) \leq 1\}$ the feasible set. The maximum operator in the function $J$ implies that $F = \{Y \mid \|Y\|_2 \leq 1\} \cap \{Y \mid \lambda^{-1}\|Y\|_\infty \leq 1\}$, hence $F$ is clearly convex, being the intersection of two convex sets. Further, the function $J(Y)$ is positive and homogenous in $Y$, hence the maximum over a linear of the objective function is achieved on the boundary $\partial F$ of the feasible set, that is, the optimum is achieved when $J(Y) = 1$.

Top solve (23), the authors in [12] propose to use a projected gradient algorithm. The gradient of the objective function $\mathbf{Tr}\, M^T Y$ is simply $M$. The projection onto the constraint set is more involved but can be treated using standard methods from convex optimization. In particular, the algorithm at each iteration $k$ involves projecting the gradient $M$ onto the tangent cone $T_F(Y_k)$ of the feasible set $F$ at the point $Y_k$. If $W_k$ is the steepest ascent direction obtained from this projection, the iterate can be updated using the (projected) gradient step

$$Y_{k+1} = \frac{Y_k + t_k W_k}{J(Y_k + t_k W_k)} \tag{24}$$

where $t_k$ is the step size that can be determined by a simple line search of the form

$$t_k = \arg\max_{t \geq 0} \left\langle M, \frac{Y_k + t W_k}{J(Y_k + t W_k)} \right\rangle \tag{25}$$

Note that the division by $J(Y_k + t_k W_k)$ ensures that the next iterate $Y_{k+1}$ lies on $\partial F$. It is shown in [12] that if the maximizing step size $t_k$ is equal to zero at some point $Y_k$, then $Y_k$ is the optimal solution to the dual problem (23).

In order to perform the projection on the tangent cone $T_F$ of $F$ at the point $Y_k$, the authors of [12] make use the following two facts:

1. For any convex subset $K \subseteq V$ of a real vector space $V$ the normal cone $N_K(x)$ at the point $x \in K$ is the polar cone to the tangent cone $T_K(x)$.

2. If two cones $C_1$ and $C_2$ in $V$ are are polar cones to each other and $\mathcal{P}_{C_1}$ and $\mathcal{P}_{C_2}$ are the projection operators onto $C_1$ and $C_2$, respectively, then $\mathcal{P}_{C_1}(X) + \mathcal{P}_{C_1}(X) = X$ for any point $X \in V$.

From these facts, it follows immediately that $\mathcal{P}_{T_F(Y_k)}(M)$, the projection of $M$ onto the dual cone of $F$ at $Y_k$, can be computed as $\mathcal{P}_{T_F(Y_k)}(M) = M - M_k$, where $M_k := \mathcal{P}_{N_F(Y_k)}(M)$. It can be shown that the normal cone is characterized by the subgradient of the function $J$ via $N(Y_k) = \{\alpha X \mid \alpha \geq 0, X \in \partial J(Y_k)\}$. Note that $J$ is in fact the pointwise maximum over two functions, hence from strong subgradient calculus it follows that

$$\partial J(Y_k) = \begin{cases} \partial \|Y_k\|_2 & \text{if } \|Y_k\|_2 > \lambda^{-1}\|Y_k\|_\infty \\ \lambda^{-1}\partial\|Y_k\|_\infty & \text{if } \|Y_k\|_2 < \lambda^{-1}\|Y_k\|_\infty \\ \text{Conv}\{\partial\|Y_k\|_2, \lambda^{-1}\partial\|Y_k\|_\infty\} & \text{if } \|Y_k\|_2 = \lambda^{-1}\|Y_k\|_\infty \end{cases} \tag{26}$$

where $\|\cdot\|_\infty$ is the maximum absolute value of the matrix entries (not the induced $\infty$-norm).

The first two cases are rather simple and the associated projections $\mathcal{P}_2(\cdot)$ and $\mathcal{P}_\infty(\cdot)$ onto the normal cones generated by the sub gradients of $\|\cdot\|_2$ and $\|\cdot\|_\infty$ at $Y_k$, respectively, are efficiently computable [12]. In particular, at each step the projection $\mathcal{P}_2(\cdot)$ requires the computation of the largest singular value[4] (and associated singular vectors) of the matrix $Y_k$, and the projection $\mathcal{P}_\infty(\cdot)$ requires only a simple element-wise comparison of the matrices $M$ and $Y_k$. The projection in the third case, i.e. when $N(Y_k) = N_2(Y_k) + N_\infty(Y_k)$, is more involved and can be performed using an alternating projections algorithm. It can be shown [12] that by initializing $S_0 = 0$ and $j = 0$, and then repeatedly setting $L_{j+1} = \mathcal{P}_2(M - S_j)$, $S_{j+1} = \mathcal{P}_\infty(M - L_{j+1})$ will yield the projection $M_k = \mathcal{P}_{N(Y_k)}(M)$ in the sense that $\lim_{j \to \infty} L_j + S_j = M_k$. The projected gradient ascent algorithm for the dual problem is given in Algortihm (4).

---

**Algorithm 4:** Projected Gradient Ascent for the Dual Problem

**Input**: Observation matrix $M$, parameter $\lambda$
initialization: $k = 0$, $Y_0 = \text{sign}(M)/J(M)$;
**while** *not converged* **do**
    Compute the projection $M_k$ of $M$ onto $N_F(Y_k)$:
    **if** $\|Y_k\|_2 > \lambda^{-1}\|Y_k\|_\infty$ **then**
        $M_k = \mathcal{P}_2(M)$, $L = M$, $S = 0$
    **else if** $\|Y_k\|_2 < \lambda^{-1}\|Y_k\|_\infty$ **then**
        $M_k = \mathcal{P}_\infty(M)$, $L = 0$, $S = M$
    **else**
        $L = 0$, $S = 0$;
        **while** *not converged* **do**
            $L = \mathcal{P}_2(M - S)$, $S = \mathcal{P}_\infty(M - L)$
        $M_k = L + S$
    Perform a line search as in (25) to determine step size $t_k$;
    $Y_{k+1} = \frac{Y_k + t_k(M - M_k)}{J(Y_k + t_k(M - M_k))}$;
    $k = k + 1$;
**Output**: $L$, $S$

---

[4]Note that the largest singular value of $Y_k$ needs to be computed anyway at each step in order to distinguish the three cases in (26)

Using the properties of dual norms, it is possible to show [12] that the primal solution $(\hat{L}, \hat{S})$ can easily be recovered from the dual optimal $\hat{Y}$. Specifically, it can be shown that if $\|Y_k\|_2 < 1$ or $\lambda^{-1}\|Y_k\|_\infty < 1$ the solution is degenerate and given by $\hat{L} = 0, \hat{S} = M$ (in case $\|Y_k\|_2 < 1$) or $\hat{L} = M, \hat{S} = 0$ (in case $\lambda^{-1}\|Y_k\|_\infty < 1$). If $\|Y_k\|_2 = \lambda^{-1}\|Y_k\|_\infty = 1$, then $(\hat{L}, \hat{S})$ is any pair of points that achieves convergence of the alternating projections method described above.

The algorithm at heart is a simple projected gradient ascent algorithm, the convergence rate of which is known to be $O(1/k)$ (outer iterations, not counting the alternating projection sub-algorithm). The most costly operations are the projections $\mathcal{P}_2(\cdot)$ that require the computation of the largest singular value of matrices of the same size as $M$.

## 2.5 Augmented Lagrangian Method

### 2.5.1 The General Case

The classic Augmented Lagrangian Method (ALM), described in [4], is a method for solving equality-constrained convex optimization problems. Consider a generic equality-constrained optimization problem of the form

$$p^* = \min_{x \in \mathcal{H}} f_o(x) \quad \text{s.t.} \quad f_c(x) = 0 \tag{27}$$

where $\mathcal{H}$ and $\mathcal{H}'$ are Hilbert spaces and $f_o : \mathcal{H} \mapsto \mathbf{R}$ and $f_c : \mathcal{H} \mapsto \mathcal{H}'$ are both convex functions. The conventional Lagrangian of the problem is $\mathcal{L}(x, \lambda) = f_o(x) + \langle \lambda, f_c(x) \rangle$, with $\lambda \in \mathcal{H}'$ being the dual variable. The augmented Lagrangian method, as its name suggests, uses an augmented Lagrangian of the form

$$\mathcal{L}(x, \lambda, \mu) = f_o(x) + \langle \lambda, f_c(x) \rangle + \frac{\mu}{2}\|f_c(x)\|^2 \tag{28}$$

where $\|\cdot\|$ is the inner product norm. Here $\lambda$ can now be interpreted as an estimate of a dual variable. The general algorithm is quite simple and given in Algorithm 5.

---

**Algorithm 5:** Generic Augmented Lagrangian Method

> **while** *not converged* **do**
> $\quad x_{k+1} = \arg\min_x \mathcal{L}(x, \lambda_k, \mu_k)$;
> $\quad \lambda_{k+1} = \lambda_k + \mu_k f_c(x_{k+1})$;
> $\quad$ update $\mu_k$ to $\mu_{k+1}$;
> $\quad k = k + 1$;
> **Output**: $X = X_k$

---

### 2.5.2 Alternating Directions ALM for Robust PCA

For the Robust PCA problem (1) the augmented Lagrangian is (associating $x = (L, S)$)

$$\mathcal{L}(L, S, Y, \mu) = \|L\|_* + \lambda\|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2}\|M - L - S\|_F^2 \tag{29}$$

Note that one of the usual assumptions made when using an augmented Lagrangian method is that the objective function $f_o$ is differentiable. This is clearly not the case for the Robust PCA problem. However, in [11] the authors show that the same convergence properties as for the differentiable case can be retained also for the Robust PCA problem. The main step in Algorithm 5 is solving the problem $x_{k+1} = \arg\min_x \mathcal{L}(x, \lambda_k, \mu_k)$, which in the case of the Robust PCA problem reads

$$(L_{k+1}, S_{k+1}) = \arg\min_{L,S} \ \|L\|_* + \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{\mu_k}{2}\|M - L - S\|_F^2 \qquad (30)$$

and can be solved using an alternating directions approach based on the ideas in section 2.2. Specifically, consider (30) for $S$ fixed. The corresponding "directional" subproblem in the variable $L$ can be formulated as

$$
\begin{aligned}
L_{k+1} &= \arg\min_L \ \|L\|_* + \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{\mu_k}{2}\|M - L - S\|_F^2 \\
&= \arg\min_L \ \|L\|_* + \langle Y_k, M - L - S\rangle + \frac{1}{2\mu_k^{-1}}\|M - L - S\|_F^2 \qquad (31)\\
&= \arg\min_L \ \|L\|_* + \frac{1}{2\mu_k^{-1}}\left\|L - (M - S + \mu_k^{-1}Y_k)\right\|_F^2
\end{aligned}
$$

Suppose $U\Sigma V^T = M - S + \mu_k^{-1}Y_k$ is the SVD of $M - S + \mu_k^{-1}Y_k$. Using the same arguments as in section 2.2, the minimizer in (31) is given by $L_{k+1} = U\mathcal{S}_{\mu_k^{-1}}[\Sigma]V^T$, where $\mathcal{S}_\varepsilon$ is the soft-thresholding operator defined in (10).

Conversely, suppose that $L$ in (30) is fixed. The "directional" subproblem in the variable $S$ is

$$
\begin{aligned}
S_{k+1} &= \arg\min_S \ \|L\|_* + \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{\mu_k}{2}\|M - L - S\|_F^2 \\
&= \arg\min_L \ \lambda\|S\|_1 + \langle Y_k, M - L - S\rangle + \frac{1}{2\mu_k^{-1}}\|M - L - S\|_F^2 \qquad (32)\\
&= \arg\min_L \ \lambda\|S\|_1 + \frac{1}{2\mu_k^{-1}}\left\|S - (M - L + \mu_k^{-1}Y_k)\right\|_F^2
\end{aligned}
$$

Again using the results from section 2.2 we find that $S_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[M - L + \mu_k^{-1}Y_k]$.

The alternating directions method for solving (30) is based on solving (31) and (32) iteratively, alternating between using a fixed $S$ to update $L$ and using the newly computed $L$ to update $S$ until convergence. Note that one can use prior iterates $L_k$ and $S_k$ for hot-starting the alternative directions based solution of (30) in order to significantly reduce the necessary number of directional steps within each iteration. These are all the ingredients needed for the alternating directions ALM method, which is given is Algorithm 6.

The following theorem gives the main convergence result for Algorithm 6.

**Theorem 3** ([11])**.** *For Algorithm 6, any accumulation point $(\hat{L}, \hat{S})$ of $(\hat{L}_k, \hat{S}_k)$ is an optimal solution to the Robust PCA problem* (1) *with convergence rate of at least $O(\mu_{k-1}^{-1})$ in the sense that*

$$\left|\|\hat{L}_k\|_* + \lambda\|\hat{S}_k\|_1 - f^*\right| = O(\mu_{k-1}^{-1})$$

*where $f^*$ is the optimal value of the Robust PCA problem.*

---

**Algorithm 6:** Alternating Directions Augmented Lagrangian Method

**Input**: Observation matrix $M$, parameter $\lambda$

initialization: $k = 0$, $\mu_0 > 0$, $\hat{Y}_0 = \text{sign}(M)/J(\text{sign}(M))$;

**while** *not converged* **do**

  $j = 0$, $L_{k+1}^0 = \hat{L}_k$, $S_{k+1}^0 = \hat{S}_k$;

  **while** *not converged* **do**

   $(U, \Sigma, V) = \text{svd}(M - S_{k+1}^j + \mu_k^{-1}\hat{Y}_k)$;

   $L_{k+1}^{j+1} = U \mathcal{S}_{\mu_k^{-1}}[\Sigma] V^T$;

   $S_{k+1}^{j+1} = \mathcal{S}_{\lambda \mu_k^{-1}}[M - L_{k+1}^{j+1} + \mu_k^{-1}\hat{Y}_k]$;

   $j = j + 1$;

  $\hat{Y}_{k+1} = \hat{Y}_k + \mu_k(M - \hat{L}_{k+1} - \hat{S}_{k+1})$;

  Update $\mu_k$ to $\mu_{k+1}$;

  $k = k + 1$;

**Output**: $L = \hat{L}_k$, $S = \hat{S}_k$

---

### 2.5.3 Inexact ALM for Robust PCA

The authors in [11] point out that it is not really necessary to solve (30) exactly at each iteration $k$ by running the alternating directions subroutine until convergence. It turns out that, under some technical conditions on the sequence $\{\mu_k\}$, one can also prove convergence of the overall algorithm when performing only a single step in either direction at each iteration $k$. This results in Algorithm 7 which in [11] is referred to as the "Inexact Augmented Lagrangian Method" (IALM).

---

**Algorithm 7:** Inexact Augmented Lagrangian Method

**Input**: Observation matrix $M$, parameter $\lambda$

initialization: $k = 0$, $\mu_0 > 0$, $Y_0 = M/J(M)$;

**while** *not converged* **do**

  $(U, \Sigma, V) = \text{svd}(M - S_k + \mu_k^{-1}Y_k)$;

  $L_{k+1} = U \mathcal{S}_{\mu_k^{-1}}[\Sigma] V^T$;

  $S_{k+1} = \mathcal{S}_{\lambda \mu_k^{-1}}[M - L_{k+1} + \mu_k^{-1}Y_k]$;

  $\hat{Y}_{k+1} = Y_k + \mu_k(M - L_{k+1} - S_{k+1})$;

  Update $\mu_k$ to $\mu_{k+1}$;

  $k = k + 1$;

**Output**: $L = \hat{L}_k$, $S = \hat{S}_k$

---

While it is possible to show convergence of Algorithm 7, the authors do not provide a bound on the rate of convergence as they do for the "exact ALM" (Algorithm 6). However, empirical results suggest that the IALM algorithm is generally significantly faster that the EALM algorithm with only slightly lower accuracy.

---

# 3 Discussion of Algorithms

The previous section discussed a number of algorithms for Robust PCA. The naive application of general-purpose interior point solvers to the dual problem works well for small problems but does not scale with the size of the data matrix $M$. The simple iterative thresholding algorithm can handle very large problem sizes but its convergence is extremely slow. Both the APG and the dual gradient ascent algorithm are much faster and better suited for large problems. An extension of the APG algorithm that uses a prediction strategy for the dimension of the principal singular space whose singular values are larger than the threshold allows to only compute a partial SVD in each step of the algorithm. With this extension the the APG algorithm turns out to be faster than the dual gradient ascent algorithm.

The current state of the art in terms of complexity, accuracy and convergence seem to be the ALM methods discussed in section 2.5. Though the authors of [11] do not give a complexity analysis for the inexact ALM algorithm, empirical results suggest that for practical problems the inexact ALM is considerably faster than its exact counterpart.

## 3.1 The Importance of the SVD

Most of the presented algorithms (in fact, all of them except for the direct use of interior point solvers on the dual) involve repeated computations of the Singular Value Decomposition (or at least a partial SVD) of matrices of considerable size. This is not very surprising, as the nuclear norm in the objective function is the sum of the singular values of the matrix argument. This repeated computation of the SVD is in fact the bottleneck of most current algorithms for Robust PCA. Hence it seems that, at least for the algorithms discussed above, being able to perform SVD on large matrices are the key to developing fast algorithms that can be used in applications with large-scale data.

When the matrix under consideration is sparse, the SVD can be carried out much faster using specialized methods [3]. Unfortunately, while the iterates of the matrix $S_k$ (in case of primal algorithms) are indeed sparse, the matrices for which the SVD actually needs to be computed are not. Therefore it does not seem clear how sparse SVD methods could help in improving performance of SVD-based Robust PCA algorithms.

If we look at the a Singular Value Thresholding operation it is obvious that we really only needs to compute those singular values that lie above the specified threshold (which is known a priori in each step and does not depend on the singular values), since the other singular values will be set to zero anyway. One possibility to speed up the algorithms that involve thresholding of singular values is therefore to use a partial SVD to compute only those singular values of interest. For the APG algorithm the authors of [11] use the software PROPACK [9] that allows the computation of such a partial SVD. However, PROPACK by default requires the dimension of the principal singular space whose singular values are larger than the threshold, which is of course unknown a priori. Since it turns out that the rank of the the iterates $L_k$ in the APG algorithm is monotonically increasing, a reasonable prediction of this dimension is not too hard [11]. Doing so then allows to a partial SVD at each step rather than a full SVD, which can potentially speed up the algorithm (if the partial SVD can be computed efficiently). The PROPACK package has also been modified to allow the computation of those singular values above a given threshold [10]. This implementation can be

used also in the ALM algorithms, where singular value thresholding is applied to matrices of the form $M - S_k + \mu_k^{-1} Y_k$.

Another potential way of speeding up many SVD-based algorithms is to exploit the fact that the matrix of which the SVD has to be computed does generally not change much between iterations, in particular after a few iterations. To utilize this fact, the authors of [13] propose a warm start technique for the block Lanczos method for SVD computation.

The following section presents a numerical comparison between Matlab's internal SVD routine, which is based on

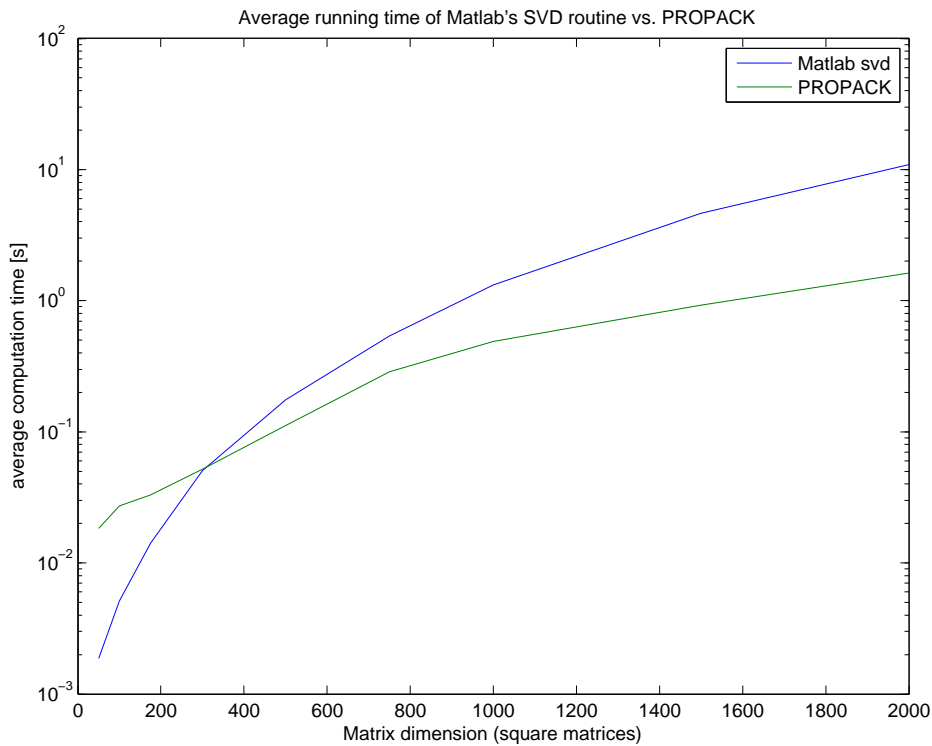### 3.1.1   Numerical comparison of different SVD methods



Figure 1: Numerical comparison of Matlab's and PROPACK's SVD routines

## 3.2   Possible Directions for Parallelization

With the advent of highly parallelized computing architectures in modern CPUs and GPUs, a number of projects have been devoted to the development and implementation of algorithms that exploit this massive computing power. Examples for these are MAGMA [16] and CULA [8], which are adapting classic high-performance linear algebra packages such as LAPACK and BLAS to the highly parallelized architecture of modern GPUs.

Some of the steps in the algorithms are inherently parallelizable, for example the entry-wise soft-thresholding of a matrix. Other steps require more work to be able to
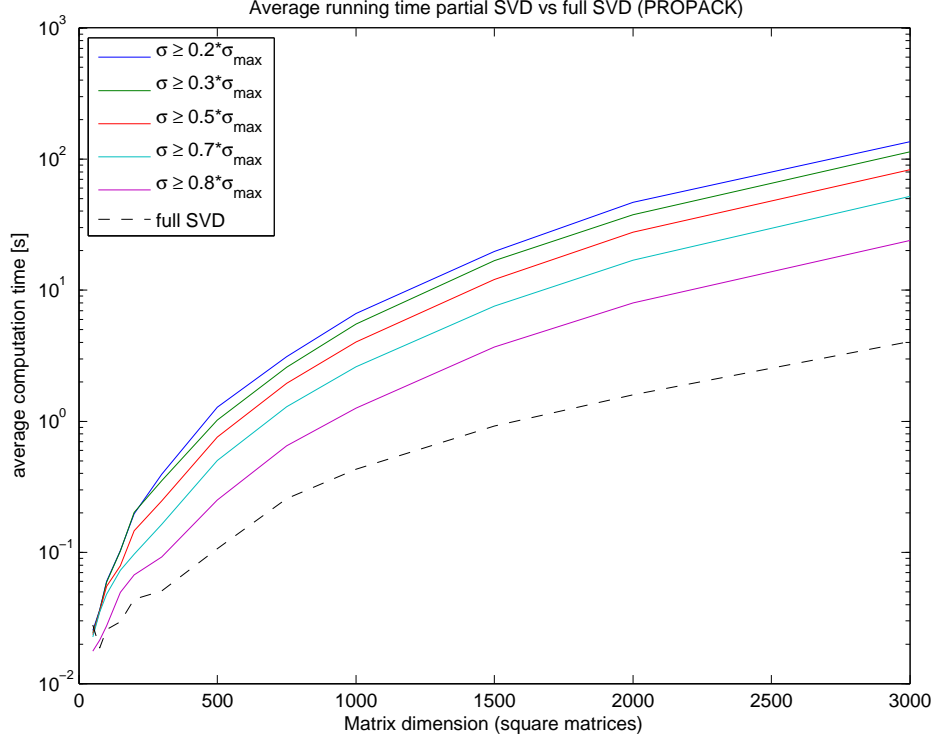
Figure 2: Numerical comparison of partial and full SVD via PROPACK

[5]

# 4 Outlook: Algorithms for Stable Principal Component Pursuit

As discussed in section REFERENCE, the results obtained for the Robust PCA problem have been extended to the case when in addition to the sparse noise component, the data is corrupted also by a small non-sparse noise component [20]. The associated optimization problem is

$$p^* = \min_{L,S} \|L\|_* + \lambda \|S\|_1$$
$$\text{s.t.} \quad \|M - L - S\|_F \leq \delta \tag{33}$$

where $\delta > 0$ is given.

[1]

# References

[1] N. S. Aybat, D. Goldfarb, and G. Iyengar. Fast First-Order Methods for Stable Principal Component Pursuit. *arXiv preprint*, 1105.2126S, May 2011.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[3] M. Berry, D. Mezher, B. Philippe, and A. Sameh. *Handbook of parallel computing and statistics*, chapter Parallel Algorithms for the Singular Value Decomposition. Statistics, textbooks and monographs. Chapman & Hall/CRC, 2005.

[4] D. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Optimization and neural computation series. Athena Scientific, 1996.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In *Foundations and Trends in Machine Learning*, volume 3, pages 1–122. 2011.

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

[7] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982, Mar. 2010.

[8] J. R. Humphrey, D. K. Price, K. E. Spagnoli, A. L. Paolini, and E. J. Kelmelis. Cula: hybrid gpu accelerated linear algebra routines. volume 7705, page 770502. SPIE, 2010.

[9] R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Dept. of Computer Science, Aarhus University, Sep 1998.

[10] Z. Lin. Some Software Packages for Partial SVD Computation. *Arxiv preprint arXiv:1108.1548*, Aug. 2011.

[11] Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *arXiv preprint*, Sept. 2010.

[12] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. M. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. Technical Report UILU-ENG-09-2214, UIUC, Jul 2009.

[13] Z. Lin and S. Wei. A Block Lanczos with Warm Start Technique for Accelerating Nuclear Norm Minimization Algorithms. *Arxiv preprint arXiv:1012.0365*, Dec. 2010.

[14] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Dokl.*, 27(2):372–376, 1983.

[15] Y. E. Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, 2007.

[16] B. J. Smith. R package magma: Matrix algebra on gpu and multicore architectures, version 0.2.2, August 2010. http://icl.cs.utk.edu/magma/index.html.

[17] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.

[18] K. C. Toh, M. J. Todd, and R. H. Tütüncü. Sdpt3 — a matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[19] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Submitted to Journal of the ACM*, 2009.

[20] Z. Zhou, X. Li, J. Wright, E. Candès, and Y. Ma. Stable principal component pursuit. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 1518 –1522, june 2010.