SMOOTH OPTIMIZATION WITH APPROXIMATE GRADIENT

ALEXANDRE D'ASPREMONT*

Abstract. We show that the optimal complexity of Nesterov's smooth first-order optimization algorithm is preserved when the gradient is only computed up to a small, uniformly bounded error. In applications of this method to semidefinite programs, this means in some instances computing only a few leading eigenvalues of the current iterate instead of a full matrix exponential, which significantly reduces the method's computational cost. This also allows sparse problems to be solved efficiently using sparse maximum eigenvalue packages.

1. Introduction.

In [13] it was shown that smooth convex minimization problems of the form:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in Q, \end{array}$$

where f is a convex function with Lipschitz continuous gradient and Q is a sufficiently simple compact convex set, could be solved with a complexity of $O(1/\sqrt{\epsilon})$, where ϵ is the precision target. Furthermore, it can be shown that this complexity bound is optimal for that class of problems (see [14] for a dicussion). More recently, [15] showed that this method could be combined with a smoothing argument to produce an $O(1/\epsilon)$ complexity bound for non-smooth problems where the objective has a saddle-function format. In particular, this meant that a broad class of semidefinite optimization problems could be solved with significantly lower memory requirements than interior point methods and a better complexity bound than classic first order methods (bundle, subgradient, etc).

Here, we show that substituting an approximate gradient, which may allow significant computation and storage savings, does not affect the optimal complexity of the algorithm in [13]. It is somewhat intuitive that an algorithm which exhibits good numerical performance in practice should be robust to at least some numerical error in the objective function and gradient computations since all implementations are necessarily computing these quantities up to some multiple of machine precision. Our objective here is to make that robustness explicit in order to design optimal schemes using only approximate gradient information.

For non-smooth problems, when the objective function $f(x)$ can be expressed as a saddle function on a compact set, the method in [15] starts by computing a smooth (i.e. with Lipschitz continuous gradient), uniform ϵ-approximation of the objective function $f(x)$, it then uses the smooth minimization algorithm in [13] to solve the approximate problem. When this smoothing technique is applied to semidefinite optimization, computing exact gradients requires forming a matrix exponential, which is often the dominant numerical step in the algorithm.

Although there are many different methods for computing this matrix exponential (see [11] for a survey), their complexity is comparable to that of a full eigenvalue decomposition of the matrix. In problem instances where only a few leading eigenvalues suffice to approximate this exponential, the per iteration complexity of the algorithm described here becomes comparable to that of classical first-order methods such as the bundle method (see [6]) or subgradient methods (see [19] for example), which have a

*ORFE Department, Princeton University, Princeton, NJ 08544. `aspremon@princeton.edu`

global complexity bound of $O(1/\epsilon^2)$ (see [14]), while keeping the optimal complexity of $O(1/\epsilon)$ of the algorithm in [15].

We apply this result to a maximum eigenvalue minimization problem (or semidefinite program with constant trace). We first recall the complexity bound derived in [16] based on a smoothing argument, using exact gradients. We produce a rough theoretical estimate of the number of eigenvalues required for convergence when approximate gradients are used. We then derive an explicit condition on the quality of the gradient approximation to guarantee convergence and compute a bound on the number of iterations. We show both on randomly generated problem instances and on problems generated from biological data sets that actual computational savings vary significantly with problem structure but can be substantial in some cases.

The paper is organized as follows. In the next section, we prove convergence of the algorithm in [13] when only an approximate gradient is used. In Section 3 we describe how these results can be applied to semidefinite optimization. Finally, in the last section we test their performance on semidefinite relaxation and maximum eigenvalue minimization problems.

2. Smooth optimization with approximate gradient.

Following the results and notations in [15, §3], we study the problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in Q, \end{array} \tag{2.1}$$

where $Q \subset \mathbf{R}^n$ is a compact convex set and f is a convex function with Lipschitz continuous gradient, such that:

$$\|\nabla f(x) - \nabla f(y)\|^* \leq L\|x-y\|, \quad x, y \in Q,$$

for some $L > 0$, which also means:

$$f(y) \leq f(x) + \langle \nabla f(x), y-x \rangle + \frac{1}{2}L\|y-x\|^2, \quad x, y \in Q. \tag{2.2}$$

The key difference here is that the *oracle* information we obtain for ∇f is *noisy*. Note that the function values are not required to compute iterates in the algorithm described here, so even if our knowledge of function values $f(x)$ is noisy, we will always use exact values in the proofs that follow. At each iteration, we obtain $\tilde{\nabla} f(x)$ satisfying:

$$|\langle \tilde{\nabla} f(x) - \nabla f(x), y-z \rangle| \leq \delta \quad x, y, z \in Q, \tag{2.3}$$

for some precision level $\delta > 0$. Throughout the paper, we assume that Q is simple enough so that this condition can be checked efficiently. As in [13], we also assume that certain projection operators on Q can be computed efficiently and we refer the reader to the end of this section for more details. Here, $d(x)$ is a prox-function for the set Q, i.e. continuous and strongly convex on Q with parameter σ (see [14] or [7] for a discussion of regularization techniques using strongly convex functions). We let x_0 be the center of Q for the prox-function $d(x)$ so that:

$$x_0 \triangleq \operatorname*{argmin}_{x \in Q} d(x),$$

assuming w.l.o.g. that $d(x_0) = 0$, we then have:

$$d(x) \geq \frac{1}{2}\sigma\|x-x_0\|^2. \tag{2.4}$$

We denote by $\tilde{T}_Q(x)$ a solution to the following subproblem:

$$\tilde{T}_Q(x) \triangleq \operatorname*{argmin}_{y\in Q}\left\{\langle\tilde{\nabla} f(x), y-x\rangle + \frac{1}{2}L\|y-x\|^2\right\} \tag{2.5}$$

We let $y_0 = \tilde{T}_Q(x_0)$ where x_0 is defined above. We recursively define three sequences of points: the current iterate $\{x_k\}$, the corresponding $y_k = \tilde{T}_Q(x_k)$, together with

$$z_k \triangleq \operatorname*{argmin}_{x\in Q}\left\{\frac{L}{\sigma}d(x) + \sum_{i=0}^{k}\alpha_i[f(x_i) + \langle\tilde{\nabla} f(x_i), x-x_i\rangle]\right\} \tag{2.6}$$

and a step size sequence $\{\alpha_k\} \geq 0$ with $\alpha_0 \in (0,1]$ so that

$$\begin{aligned} x_{k+1} &= \tau_k z_k + (1-\tau_k)y_k \\ y_{k+1} &= \tilde{T}_Q(x_{k+1}) \end{aligned} \tag{2.7}$$

where $\tau_k = \alpha_{k+1}/A_{k+1}$ with $A_k = \sum_{i=0}^{k}\alpha_i$. We implicitly assume here that the two subproblems defining y_k and z_k can be solved very efficiently (in the examples that follow, they amount to Euclidean projections). We will show recursively that for a good choice of step sequence α_k, the iterates x_k and y_k satisfy the following relationship (denoted by $\mathcal{R}_k$):

$$A_k f(y_k) \leq \psi_k + A_k g(k,\delta) \quad (\mathcal{R}_k)$$

where $g(k,\delta)$ measures the accumulated gradient approximation error and will be bounded in Lemma 2.1, and

$$\psi_k \triangleq \min_{x\in Q}\left\{\frac{L}{\sigma}d(x) + \sum_{i=0}^{k}\alpha_i[f(x_i) + \langle\tilde{\nabla} f(x_i), x-x_i\rangle]\right\}.$$

First, using $d(x) \geq \frac{1}{2}\sigma\|x-x_0\|^2$, then inequality (2.2) and condition (2.3), we have:

$$\psi_0 = \min_{x\in Q}\left\{\frac{L}{\sigma}d(x) + \alpha_0[f(x_0) + \langle\tilde{\nabla} f(x_0), x-x_0\rangle]\right\} \geq \alpha_0 f(y_0) - \alpha_0\delta$$

which is $\mathcal{R}_0$. We can then bound the approximation error in the following result.

Lemma 2.1. *Let α_k be a step sequence satisfying:*

$$0 < \alpha_0 \leq 1 \quad \text{and} \quad \alpha_k^2 \leq A_k, \quad k \geq 0, \tag{2.8}$$

suppose that $(\mathcal{R}_k)$ holds with x_{k+1} and y_{k+1} are defined as in (2.7), then $(\mathcal{R}_{k+1})$ holds with:

$$g(k+1,\delta) = (1-\tau_k)g(k,\delta) + \tau_k 3\delta,$$

where $\tau_k \in [0,1]$ and $g(0,\delta) = \alpha_0\delta$.

Proof. Let us assume that $(\mathcal{R}_k)$ holds. Because $d(x)$ is strongly convex with parameter σ, the function:

$$\frac{L}{\sigma}d(x) + \sum_{i=0}^{k}\alpha_i[f(x_i) + \langle\tilde{\nabla} f(x_i), x-x_i\rangle]$$

is strongly convex with parameter L. Using this property and the definition of z_k we obtain:

$$\begin{aligned}
\psi_{k+1} &= \min_{x\in Q}\left\{\frac{L}{\sigma}d(x) + \sum_{i=0}^{k+1}\alpha_i[f(x_i) + \langle\tilde{\nabla}f(x_i), x - x_i\rangle]\right\}\\
&\geq \min_{x\in Q}\left\{\psi_k + \frac{1}{2}L\|x - z_k\|^2 + \alpha_{k+1}[f(x_{k+1}) + \langle\tilde{\nabla}f(x_{k+1}), x - x_{k+1}\rangle]\right\}.
\end{aligned}$$

Now, using $(\mathcal{R}_k)$ then the convexity of $f(x)$, we get:

$$\begin{aligned}
&\psi_k + A_k g(k,\delta) + \alpha_{k+1}[f(x_{k+1}) + \langle\tilde{\nabla}f(x_{k+1}), x - x_{k+1}\rangle]\\
&\geq A_k f(y_k) + \alpha_{k+1}[f(x_{k+1}) + \langle\tilde{\nabla}f(x_{k+1}), x - x_{k+1}\rangle]\\
&\geq A_k[f(x_{k+1}) + \langle\nabla f(x_{k+1}), y_k - x_{k+1}\rangle] + \alpha_{k+1}[f(x_{k+1}) + \langle\tilde{\nabla}f(x_{k+1}), x - x_{k+1}\rangle],
\end{aligned}$$

and condition (2.3), together with (2.7) imply that:

$$\begin{aligned}
&A_k[f(x_{k+1}) + \langle\nabla f(x_{k+1}), y_k - x_{k+1}\rangle] + \alpha_{k+1}[f(x_{k+1}) + \langle\tilde{\nabla}f(x_{k+1}), x - x_{k+1}\rangle]\\
&\geq A_{k+1}f(x_{k+1}) + \langle\nabla f(x_{k+1}), A_k y_k - A_k x_{k+1} + \alpha_{k+1}(x - x_{k+1})\rangle - \alpha_{k+1}\delta\\
&= A_{k+1}f(x_{k+1}) + \alpha_{k+1}\langle\nabla f(x_{k+1}), x - z_k\rangle - \alpha_{k+1}\delta.
\end{aligned}$$

Because α_k satisfies (2.8), we have $\tau_k^2 \leq A_{k+1}^{-1}$ and can combine the last three inequalities to get:

$$\begin{aligned}
\psi_{k+1} \geq\ & A_{k+1}f(x_{k+1}) - A_k g(k,\delta) - \alpha_{k+1}\delta\\
& + \min_{x\in Q}\left\{\tfrac{1}{2}L\|x - z_k\|^2 + \alpha_{k+1}\langle\nabla f(x_{k+1}), x - z_k\rangle\right\}\\
\geq\ & A_{k+1}[f(x_{k+1}) - (1-\tau_k)g(k,\delta) - \tau_k\delta\\
& + \min_{x\in Q}\left\{\tfrac{1}{2}L\tau_k^2\|x - z_k\|^2 + \tau_k\langle\nabla f(x_{k+1}), x - z_k\rangle\right\}].
\end{aligned}\tag{2.9}$$

Let us define $y \triangleq \tau_k x + (1-\tau_k)y_k$ so that $y - x_{k+1} = \tau_k(x - z_k)$, with:

$$\begin{aligned}
&\min_{x\in Q}\left\{\tfrac{1}{2}L\tau_k^2\|x - z_k\|^2 + \tau_k\langle\nabla f(x_{k+1}), x - z_k\rangle\right\}\\
&= \min_{\{y\in\tau_k Q + (1-\tau_k)y_k\}}\left\{\tfrac{1}{2}L\|y - x_{k+1}\|^2 + \langle\nabla f(x_{k+1}), y - x_{k+1}\rangle\right\}.
\end{aligned}$$

Combining condition (2.3) with the fact that $y - x_{k+1} = \tau_k(x - z_k)$ for some $x, z_k \in Q$, we get:

$$\begin{aligned}
&\min_{\{y\in\tau_k Q + (1-\tau_k)y_k\}}\left\{\tfrac{1}{2}L\|y - x_{k+1}\|^2 + \langle\nabla f(x_{k+1}), y - x_{k+1}\rangle\right\}\\
&\geq \min_{\{y\in\tau_k Q + (1-\tau_k)y_k\}}\left\{\tfrac{1}{2}L\|y - x_{k+1}\|^2 + \langle\tilde{\nabla}f(x_{k+1}), y - x_{k+1}\rangle\right\} - \tau_k\delta
\end{aligned}$$

Now, because Q is convex, we must have $\tau_k Q + (1-\tau_k)y_k \subset Q$ and:

$$\begin{aligned}
&\min_{\{y\in\tau_k Q + (1-\tau_k)y_k\}}\left\{\tfrac{1}{2}L\|y - x_{k+1}\|^2 + \langle\tilde{\nabla}f(x_{k+1}), y - x_{k+1}\rangle\right\} - \tau_k\delta\\
&\geq \min_{y\in Q}\left\{\tfrac{1}{2}L\|y - x_{k+1}\|^2 + \langle\tilde{\nabla}f(x_{k+1}), y - x_{k+1}\rangle\right\} - \tau_k\delta.
\end{aligned}$$

By the definition of $y_{k+1} = \tilde{T}_Q(x_{k+1})$ and using condition (2.3), we get:

$$\begin{aligned}
&\min_{y\in Q}\left\{\tfrac{1}{2}L\|y-x_{k+1}\|^2+\langle\tilde{\nabla}f(x_{k+1}),y-x_{k+1}\rangle\right\}-\tau_k\delta\\
&=\tfrac{1}{2}L\|\tilde{T}_Q(x_{k+1})-x_{k+1}\|^2+\langle\tilde{\nabla}f(x_{k+1}),\tilde{T}_Q(x_{k+1})-x_{k+1}\rangle-\tau_k\delta\\
&\geq\tfrac{1}{2}L\|y_{k+1}-x_{k+1}\|^2+\langle\nabla f(x_{k+1}),y_{k+1}-x_{k+1}\rangle-2\tau_k\delta,
\end{aligned}$$

and inequality (2.2) gives:

$$\begin{aligned}
&\tfrac{1}{2}L\|y_{k+1}-x_{k+1}\|^2+\langle\nabla f(x_{k+1}),y_{k+1}-x_{k+1}\rangle-2\tau_k\delta,\\
&\geq f(y_{k+1})-f(x_{k+1})-2\tau_k\delta.
\end{aligned}$$

Combining these inequalities with the inequality on ψ_{k+1} in (2.9), we finally get:

$$\psi_{k+1}\geq A_{k+1}\left[f(y_{k+1})-(1-\tau_k)g(k,\delta)-3\tau_k\delta\right]$$

which is the desired result. □

We can use this result to study the convergence of the following algorithm given only approximate gradient information.

Smooth minimization with approximate gradient.

```
Starting from $x_0$, the prox center of the set $Q$, we iterate:
 1. compute $\tilde{\nabla}f(x_k)$,
 2. compute $y_k = \tilde{T}_Q(x_k)$,
 3. compute $z_k = \mathrm{argmin}_{x\in Q}\left\{\frac{L}{\sigma}d(x)+\sum_{i=0}^{k}\alpha_i[f(x_i)+\langle\tilde{\nabla}f(x_i),x-x_i\rangle]\right\}$,
 4. update $x$ using $x_{k+1} = \tau_k z_k + (1-\tau_k)y_k$,
```

Again, because solving for y_k and z_k can often be done very efficiently, the dominant numerical step in this algorithm is the evaluation of $\tilde{\nabla}f(x_k)$. If the step size sequence α_k satisfies the conditions of Lemma 2.1, we can show the following convergence result:

Theorem 2.2. *Suppose α_k satisfies equation (2.8), with the iterates x_k and y_k defined in (2.6) and (2.7), then for any $k \geq 0$ we have:*

$$f(y_k)-f(x^\star)\leq\frac{Ld(x^\star)}{A_k\sigma}+3\delta$$

where $x^\star$ is an optimal solution to problem (2.1).

Proof. If α_k satisfies the hypotheses of lemma 2.1 we have:

$$A_kf(y_k)\leq\psi_k+A_kg(k,\delta)$$

where $A_k=\sum_{i=0}^{k}\alpha_i$ and $g(k,\delta)\leq 3\delta$. Now, because $f(x)$ is convex, we also have:

$$\psi_k\leq\frac{L}{\sigma}d(x^\star)+A_kf(x^\star)+A_k3\delta$$

which yields the desired result. □

When $d(x^\star) < +\infty$ (e.g. if Q is bounded), if we set the step sequence as $\alpha_k = (k+1)/2$ and δ to some fraction of the target precision ϵ (here $\epsilon/6$), A_k grows as $O(k^2)$ and Theorem 2.2 ensures that the algorithm will converge to an ϵ solution in less than:

$$\sqrt{\frac{8Ld(x^\star)}{\sigma\epsilon}} \tag{2.10}$$

iterations. In practice of course, $d(x^\star)$ needs to be bounded a priori and L is often hard to evaluate. A notable exception is when $f(x)$ is a smooth approximation (as in [15, 16] for example), in which case L is know explicitly as a function of the precision. We have implicitly assumed, as in [13], that the set Q is simple enough so that the complexity of solving the two minimization subproblems in steps 2 and 3 of the algorithm is low relative to that of approximating the gradient. We also implicitly assumed that the set Q is simple enough so that condition (2.3) can be checked efficiently. In the numerical experiments of Section 4 for example, steps 2 and 3 are Euclidean projections on the unit box and condition (2.3) is a simple inequality on the leading eigenvalues of the current iterate.

3. Semidefinite optimization.

Here, we describe in detail how the results of the previous section can be applied to semidefinite optimization. We consider the following maximum eigenvalue problem:

$$\begin{array}{ll} \text{minimize} & \lambda^{\max}(A^Ty+c) - b^Ty \\ \text{subject to} & y \in Q, \end{array} \tag{3.1}$$

in the variable $y \in \mathbf{R}^m$, with parameters $A \in \mathbf{R}^{m\times n^2}$, $b \in \mathbf{R}^m$ and $c \in \mathbf{R}^{n^2}$. Let us remark that when Q is equal to $\mathbf{R}^m$, the dual of this program is a semidefinite program with constant trace written:

$$\begin{array}{ll} \text{maximize} & c^Tx \\ \text{subject to} & Ax = b \\ & \mathbf{Tr}(x) = 1 \\ & x \succeq 0, \end{array} \tag{3.2}$$

in the variable $x \in \mathbf{R}^{n^2}$, where $\mathbf{Tr}(x) = 1$ means that the matrix obtained by reshaping the vector x has trace equal to one and $x \succeq 0$ means that this same matrix is symmetric, positive semidefinite.

3.1. Smoothing technique.

As in [12], [16], [3] or [2] we can find a uniform ϵ-approximation to $\lambda^{\max}(X)$ with Lipschitz continuous gradient. Let $\mu > 0$ and $X \in \mathbf{S}_n$, we define:

$$f_\mu(X) = \mu \log\left(\sum_{i=1}^n e^{\lambda_i(X)/\mu}\right) = \mu\log\left(e^{\frac{\lambda^{\max}(X)}{\mu}}\left(1+\sum_{i=2}^n e^{\frac{\lambda_i(X)-\lambda^{\max}(X)}{\mu}}\right)\right)$$

where $\lambda_i(X)$ is the ith eigenvalue of X. This is also:

$$f_\mu(X) = \lambda^{\max}(X) + \mu\log\mathbf{Tr}\left(\exp\left(\frac{X-\lambda^{\max}(X)\mathbf{I}}{\mu}\right)\right) \tag{3.3}$$

which requires computing a matrix exponential at a numerical cost of $O(n^3)$. We then have:

$$\lambda^{\max}(X) \le f_\mu(X) \le \lambda^{\max}(X) + \mu\log n,$$

so if we set $\mu = \epsilon / \log n$, $f_\mu(X)$ becomes a uniform ϵ-approximation of $\lambda^{\max}(X)$. In [16] it was shown that $f_\mu(X)$ has a Lipschitz continuous gradient with constant:

$$L = \frac{1}{\mu} = \frac{\log n}{\epsilon}.$$

The gradient $\nabla f_\mu(X)$ can also be computed explicitly as:

$$\frac{\exp\left(\frac{X-\lambda^{\max}(X)\mathbf{I}}{\mu}\right)}{\mathbf{Tr}\left(\exp\left(\frac{X-\lambda^{\max}(X)\mathbf{I}}{\mu}\right)\right)} \tag{3.4}$$

using the same matrix exponential as in (3.3). Let $\|y\|$ be some norm on $\mathbf{R}^m$ and $d(x)$ a strongly convex prox-function with parameter $\sigma > 0$. As in [16], we define:

$$\|A\| = \max_{\|h\|=1} \|A^T h\|_2,$$

where $\|A^T h\|_2 = \max_i |\lambda_i(A^T h)|$. The algorithm detailed in [15], where *exact* function values and gradients are computed, will find an ϵ solution to (3.1) after at most:

$$\frac{2\|A\|}{\epsilon}\sqrt{\frac{\log n}{\sigma} d(y^\star)} \tag{3.5}$$

iterations, each iteration requiring a matrix exponential computation.

3.2. Spectrum & expected performance gains. The complexity estimate above is valid when the matrix exponential in (3.3) is computed exactly, at a cost of $O(n^3)$. As we will see below, only a few leading eigenvalues are sometimes required to satisfy conditions (2.3) and obtain a comparable complexity estimate at a much lower numerical cost. To illustrate the potential complexity gains, let us pick a matrix $X \in \mathbf{S}_n$ whose coefficients are centered independent normal variables with second moment given by σ^2/n. From Wigner's semicircle law, $\lambda^{\max}(X) \sim 2\sigma$ as n goes to infinity and the eigenvalues of X are asymptotically distributed according to the density:

$$p(x) = \frac{1}{2\pi\sigma^2}\sqrt{4\sigma^2 - x^2},$$

which means that, in the limit, the proportion of eigenvalues required to reach a precision of γ in the exponential is given by:

$$P_\lambda \triangleq P\left(e^{\frac{\lambda_i(X)-\lambda^{\max}(X)}{\mu}} \leq \gamma\right) = \int_{-2\sigma}^{2\sigma+\epsilon\frac{\log\gamma}{\log n}} \frac{1}{2\pi\sigma^2}\sqrt{4\sigma^2 - x^2}dx.$$

Since the problems under consideration are relaxations of sparse PCA, we can also consider the case where $X \in \mathbf{S}_n$ is sampled from the Wishart distribution. In that case, the eigenvalues are distributed according to the Marčenko-Pastur distribution (see [10]) and the above proportion becomes:

$$P_\lambda = P\left(e^{\frac{\lambda_i(X)-\lambda^{\max}(X)}{\mu}} \leq \gamma\right) = \int_{-2\sigma}^{2\sigma+\epsilon\frac{\log\gamma}{\log n}} \frac{\sqrt{x(4\sigma - x)}}{2\pi x}dx.$$

With $n = 5000$, $\gamma = 10^{-6}$ and $\epsilon = 10^{-2}$, we get $nP_\lambda = 2.3$, so the approximations above would suggest that, in theory, it is only necessary to compute about three eigenvalues per iteration to get an approximation with precision $\gamma = 10^{-6}$. In practice however, the results of Section 4 show that these rough estimates should be significantly tempered.

3.3. Global complexity bound. Let us now focus on the following program:

$$\begin{array}{ll} \text{minimize} & \lambda^{\max}(A^T y + c) \\ \text{subject to} & \|y\| \leq \beta, \end{array} \tag{3.6}$$

where the set Q is here explicitly given by :

$$Q = \{y \in \mathbf{R}^p : \ \|y\| \leq \beta\},$$

for some $\beta > 0$ with $\|.\|$ the Euclidean norm here. We can pick $\|x\|^2/2$ as a prox function for Q, which is strongly convex with convexity parameter 1. Let $\lambda(X) \in \mathbf{R}^n$ be the eigenvalues of the matrix $X = A^T y + c$, in decreasing order, with $u_i(X) \in \mathbf{R}^n$ an orthonormal set of eigenvectors. The gradient matrix of $\exp(X/\mu)$ is written:

$$\nabla f_\mu(X) = \left(\sum_{i=1}^n e^{\frac{\lambda_i(X)}{\mu}}\right)^{-1} \sum_{i=1}^n e^{\frac{\lambda_i(X)}{\mu}} u_i(X) u_i(X)^T,$$

Suppose we only compute the first m eigenvalues and use them to approximate this gradient by:

$$\tilde{\nabla} f_\mu(X) = \left(\sum_{i=1}^m e^{\frac{\lambda_i(X)}{\mu}}\right)^{-1} \sum_{i=1}^m e^{\frac{\lambda_i(X)}{\mu}} u_i(X) u_i(X)^T,$$

we get the following bound on the error:

$$\|\nabla f_\mu(X) - \tilde{\nabla} f_\mu(X)\| \leq \frac{\sqrt{2}(n-m) e^{\frac{\lambda_m(X) - \lambda_1(X)}{\mu}}}{\left(\sum_{i=1}^m e^{\frac{\lambda_i(X)-\lambda_1(X)}{\mu}}\right)}.$$

In this case, with $X = A^T y - c$ here, condition (2.3) means that we only need to compute m eigenvalues with m such that:

$$\frac{\sqrt{2}(n-m) e^{\frac{\lambda_m(X) - \lambda_1(X)}{\mu}}}{\left(\sum_{i=1}^m e^{\frac{\lambda_i(X)-\lambda_1(X)}{\mu}}\right)} \leq \frac{\delta}{\sigma^{\max}(A)}, \tag{3.7}$$

where $\sigma^{\max}(A)$ is the largest singular value of the matrix A. Using the result in [16], if we define $\|A\| = \max_{\|h\|=1} \|Ah\|_2$ and set $\delta = \epsilon/6$, the algorithm in Section 2 will then converge to an ϵ-solution of problem (3.6) in at most:

$$\frac{4\|A\|\beta}{\epsilon}\sqrt{\log n} \tag{3.8}$$

iterations. This bound on the number of iterations is independent of m in condition (3.7), i.e. the number of eigenvalues required at each iteration. The cost per iteration however varies with problem structure as each iteration requires computing m leading eigenvalues, which can be performed in $O(mn^2)$ operations. Note that partial eigenvalue decompositions only access the matrix through matrix-vector products (see [8]), hence can handle sparse problems very efficiently. The threshold δ can be adjusted empirically to tradeoff between the number of iterations and the numerical cost of each iteration. Unfortunately, we can't directly infer a bound on m from the structure of A, so in the next section we study the link between m and the matrix spectrum in numerical examples.

4. Examples & numerical performance. In this section, we illustrate the behavior of the approximate gradient algorithm on various semidefinite optimization problems. Overall, while there appears to be a direct link between problem structure and complexity (i.e. the number of eigenvalues required in the gradient approximation) in the first sparse PCA example discussed below, we will observe on random maximum eigenvalue minimization problems that predicting complexity based on overall problem structure remains an open numerical question in general.

4.1. Sparse principal component analysis. Based on the results in [3], the problem of finding a sparse leading eigenvector of a matrix $C \in \mathbf{S}_n$ can be written:

$$\begin{array}{ll} \text{maximize} & x^T C x \\ \text{subject to} & \|x\|_2 = 1 \\ & \mathbf{Card}(x) \leq k, \end{array} \tag{4.1}$$

where $\mathbf{Card}(x)$ is the number of nonzero coefficients in x, and admits the following semidefinite relaxation:

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(CX) - \rho \mathbf{1}^T |X| \mathbf{1} \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & X \succeq 0, \end{array} \tag{4.2}$$

which is a semidefinite program in the variable $X \in \mathbf{S}^n$, where $\rho > 0$ is the penalty controlling the sparsity of the solution. Its dual is given by:

$$\begin{array}{ll} \text{minimize} & \lambda^{\max}(C + U) \\ \text{subject to} & |U_{ij}| \leq \rho, \quad i, j = 1, \ldots, n, \end{array} \tag{4.3}$$

which is of the form (3.1) with

$$Q = \{U \in \mathbf{S}_n : \; |U_{ij}| \leq \rho, \; i, j = 1, \ldots, n\}.$$

The smooth algorithm detailed in Section 2 is explicitly described for this problem in [3] and implemented in a numerical package called DSPCA which we have used in the examples here. To test its performance, we generate a matrix M with uniformly distributed coefficients in $[0, 1]$. We let $e \in \mathbf{R}^{250}$ be a sparse vector with:

$$e = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, \ldots).$$

We then form a test matrix $C = M^T M + vee^T$, where v is a signal-to-noise ratio.

In Figure 4.1 on the left, we plot duality gap versus CPU time used for values of the signal to noise ratio v ranging from 10 to 100. In Figure 4.1 on the right, we plot number of eigenvalues required against computing time using a covariance matrix of dimension $n = 500$ sampled from the colon cancer data set in [1], and a noisy rank one matrix. Finally, we measure total computing time versus problem dimension n on this same data set, by solving problem (4.2) for increasingly large submatrices of the original covariance matrix. In each of these examples, we stop after the duality gap has been reduced by 10^{-2}, which is enough here to identify sparse principal components. In Figure 4.2 on the left, we plot computing time versus target precision in loglog scale, on a sparse PCA problem of size 200 extracted from the colon cancer data set. In the previous section, we have seen that precision impacts computing time both through the total number of iterations in (3.5) and through condition (3.7) on the number of eigenvalues required in the gradient approximation. In this example, we

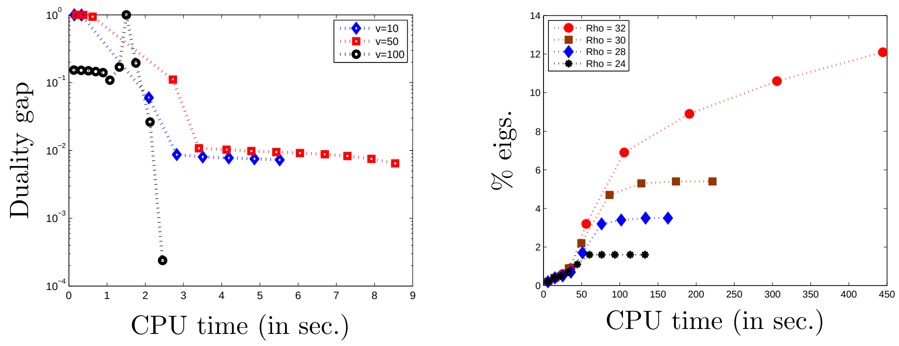

Fig. 4.1. *Left: Duality gap versus CPU time for various values of the signal to noise ratio v. Right: Percentage of eigenvalues required versus CPU time, for various values of the penalty parameter ρ controlling sparsity.*

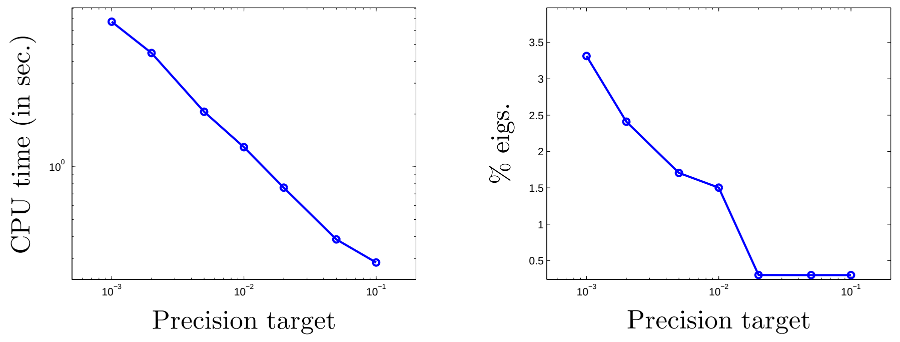

Fig. 4.2. *Left: CPU time (in seconds) versus target precision in loglog scale. Right: average percentage of eigenvalues required at each iteration versus target precision, in semilog scale.*

observe that CPU time increases a little bit slower than the upper bound of $O(1/\epsilon)$ given in (4.2). In Figure 4.2 on the right, we plot the average percentage of eigenvalues required at each iteration versus target precision, in semilog scale. We observe, on this example of dimension 200, that for low target precisions, one eigenvalue is often enough to approximate the gradient, but that this number quickly increases for higher precision targets. Note that in all cases, the precision targets are significantly lower than those achieved by interior point methods (usually at least 10^{-8}) but the cost per iteration and storage requirements of the first-order algorithms detailed here are also significantly lower.

In Table 4.1, we then compare total CPU time using a full precision matrix exponential, against CPU time using only a partial eigenvalue decomposition to approximate this exponential. Note that other classic methods for computing the matrix exponential such as Padé approximations (see [11]), did not provide a significant performance benefit and are not included here. Both exact and approximate gradient codes are fully written in C, with partial eigenvalue decompositions computed using the FORTRAN package ARPACK (see [8]) with calls to vendor optimized BLAS and LAPACK for matrix operations. To improve stability, the size of the Lanczos basis in ARPACK was set at four times the number of eigenvectors required. We observe that, on these problems, the partial eigenvalue decomposition method is about ten times faster.

	$n = 100$	$n = 200$	$n = 500$
Rank one, Full	3.2	8.0	14.7
Rank one, **Partial**	0.4	0.75	1.6
Colon, Full	2.6	18.1	274.3
Colon, **Partial**	0.3	1.3	17.7

TABLE 4.1

CPU time (in seconds) versus problem dimension n for full and partial eigenvalue matrix exponential computations.

4.2. Matrix structure and complexity: open numerical issues. The previous section showed how the spectrum of the current iterate impacts the complexity of the algorithm detailed in this paper: a steeply decreasing spectrum allows fewer eigenvalues to be computed in the matrix exponential approximation, and a wider gap between eigenvalues improves the convergence rate of these eigenvalue computations. In this section, we study the number of eigenvalues required in randomly generated maximum eigenvalue minimization problems. Because of the measure concentration phenomenon, there is nothing really random about the spectrum of large-scale, naively generated semidefinite optimization problems so we begin by detailing a simple method for generating random matrices with a given spectrum.

Generating random matrices with a given spectrum. Suppose $X \in \mathbf{S}_n$ is a matrix with normally distributed coefficients, $X_{ij} \sim \mathcal{N}(0,1)$, $i,j = 1,\ldots,n$. If we write its QR decomposition, $X = QR$ with $Q,\ R \in \mathbf{R}^{n\times n}$, then the orthogonal matrix Q is Haar distributed on the orthogonal group $\mathcal{O}_n$ (see [4] for example). This means that to generate a random matrix with given spectrum $\lambda \in \mathbf{R}^n$, we generate a normally distributed matrix X, compute its QR decomposition and the matrix $Q\,\mathrm{diag}(\lambda)Q^T$ will be uniformly distributed on the set of symmetric matrices with spectrum λ.

Maximum eigenvalue minimization. We now form random maximum eigenvalue minimization problems, then study how the number of required eigenvalues in the gradient computation evolves as the solution approaches optimality. We solve the following problem:

$$\begin{array}{ll} \text{minimize} & \lambda^{\max}(A^T y + c) \\ \text{subject to} & \|y\| \leq \beta, \end{array}$$

in the variable $y \in \mathbf{R}^m$, where $c \in \mathbf{R}^{n^2}$, $A \in \mathbf{R}^{m\times n^2}$ and $\beta > 0$ is an upper bound on the norm of the solution. In Figure 4.3 we plot percentage of eigenvalues required in the gradient computation versus duality gap for randomly generated problem instances where $n = 50$ and $m = 25$. The first two plots use data matrices with Gaussian and Wishart distributions, whose spectrum are distributed according to Wigner's semicircle law and the Marčenko-Pastur distribution respectively. The last two plots use the procedure described above to generate matrices with uniform spectrum on $[0,1]$, and uniform spectrum on $[0,1]$ with one eigenvalue set to 5. We observe that the number of eigenvalues required in the algorithm varies significantly with matrix spectrum.

Problem structure and effective complexity. The results on sparse PCA in §4.1 and on the random problems of this section show that problem structure has a significant impact on performance. Predicting how many eigenvalues will be required at each iteration based on structural properties of the problem is an important but difficult question. In particular, the number of eigenvalues required in the Gaussian case is much higher than what the asymptotic analysis in Section 3.2 predicted.

Furthermore, in the sparse PCA example, complexity seems to vary with problem structure somewhat intuitively: higher signal to noise ratio means lower complexity and a higher sparsity target means higher complexity. However, this is not the case in the random problems studied here, two unstructured problems (uniform and Wishart) have low complexity while one requires computing many more eigenvalues per iteration (Gaussian) and a more structured example (uniform plus rank one) also requires many eigenvalues. Overall then, predicting effective complexity (i.e. the number of eigenvalues required at each iteration) based on problem structure remains a difficult open question at this point.

Also, it is well known empirically (see [17], [9] and [18] among others) that the largest eigenvalues of $A^T y - c$ in (3.1) tend to coalesce near the optimum, thus potentially increasing the number of eigenvalues required when computing $\tilde{\nabla} f(x)$ and the number of iterations required for computing leading eigenvalues (see [5] for example), but in these references too, no a priori link between coalescence and problem structure is established. This coalescence phenomenon is never apparent in the numerical examples studied here, perhaps because it only appears at the much higher precision targets reached by interior point methods.

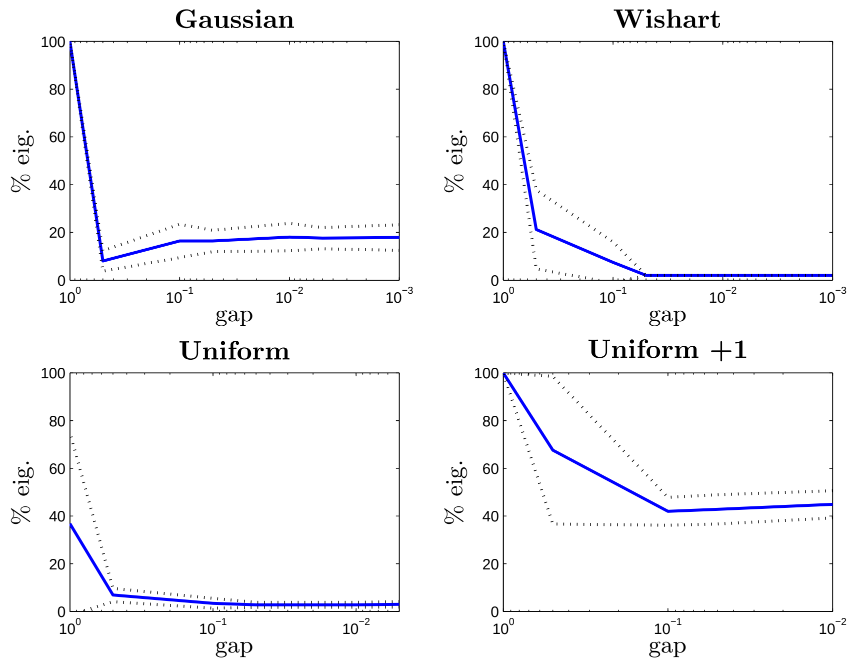

Fig. 4.3. *Average percentage of eigenvalues required (solid line) versus duality gap on randomly generated maximum eigenvalue minimization problems, for various problem matrix distributions. Dashed lines at plus and minus one standard deviation.*

Acknowledgements. The author would like to thank Noureddine El Karoui for very useful comments and to acknowledge support from NSF grant DMS-0625352, ONR grant number N00014-07-1-0150, a Peek junior faculty fellowship and a gift from Google, Inc.

REFERENCES

[1] A. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays*, Cell Biology, 96 (1999), pp. 6745–6750.

[2] A. Ben-Tal and A. Nemirovski, *Non-Euclidean restricted memory level method for large-scale convex optimization*, Mathematical Programming, 102 (2005), pp. 407–456.

[3] A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. R. G. Lanckriet, *A direct formulation for sparse PCA using semidefinite programming*, SIAM Review, 49 (2007), pp. 434–448.

[4] P. Diaconis, *Patterns in eigenvalues: The 70th Josiah Willard Gibbs lecture*, Bulletin of the American Mathematical Society, 40 (2003), pp. 155–178.

[5] G.H. Golub and C.F. Van Loan, *Matrix computation*, North Oxford Academic, (1990).

[6] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM Journal on Optimization, 10 (2000), pp. 673–696.

[7] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal, *Convex Analysis and Minimization Algorithms*, Springer, 1993.

[8] R.B. Lehoucq, D.C. Sorensen, and C. Yang, *ARPACK: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Society for Industrial & Applied Mathematics, 1998.

[9] A.S. Lewis and M.L. Overton, *Eigenvalue optimization*, Acta Numerica, 5 (1996), pp. 149–190.

[10] V.A. Marčenko and L.A. Pastur, *Distribution of eigenvalues for some sets of random matrices*, Mathematics of the USSR - Sbornik, 1 (1967), pp. 457–483.

[11] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review, 45 (2003), pp. 3–49.

[12] A. Nemirovski, *Prox-method with rate of convergence O(1/T) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems*, SIAM Journal on Optimization, 15 (2004), pp. 229–251.

[13] Y. Nesterov, *A method of solving a convex programming problem with convergence rate* $O(1/k^2)$, Soviet Mathematics Doklady, 27 (1983), pp. 372–376.

[14] ———, *Introductory Lectures on Convex Optimization*, Springer, 2003.

[15] ———, *Smooth minimization of non-smooth functions*, Mathematical Programming, 103 (2005), pp. 127–152.

[16] ———, *Smoothing technique and its applications in semidefinite optimization*, Mathematical Programming, 110 (2007), pp. 245–259.

[17] M. L. Overton, *Large scale optimization of eigenvalues*, SIAM Journal on Optimization, 2 (1992), pp. 88–120.

[18] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Mathematics of Operations Research, 23 (1998), pp. 339–358.

[19] N. Z. Shor, *Minimization Methods for Non-differentiable Functions*, Springer-Verlag, Berlin, 1985.