CCIT Report #559

October 2005

Sequential Subspace Optimization Method for Large-Scale Unconstrained Problems

Guy Narkiss and Michael Zibulevsky

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel.

October 5, 2005

Abstract

We present the Sequential Subspace Optimization (SESOP) method for large-scale smooth unconstrained problems. At each iteration we search for a minimum of the objective function over a subspace spanned by the current gradient and by directions of few previous steps. We also include into this subspace the direction from the starting point to the current point, and a weighted sum of all previous gradients, following [Nemirovski-1982]. This safeguard measure provides an optimal worst case convergence rate of order $1/N^2$ (for convex problems), where N is the iteration count. In the case of quadratic objective, the method is equivalent to the conjugate gradients method.

We identify an important class of problems, where subspace optimization can be implemented extremely fast. This happens when the objective function is a combination of expensive linear mappings with computationally cheap non-linear functions. This is a typical situation in many applications, like tomography, signal and image denoising with Basis Pursuit, pattern recognition with Support Vector Machine, and many others. We demonstrate highly competitive numerical results using examples from the mentioned areas.

1 Introduction

We consider an unconstrained minimization of a smooth function

$$\min_{\mathbf{x}\in\mathcal{R}^n} f(\mathbf{x}). \tag{1}$$

When the number of variables is very large, say $n = 10^4 - 10^7$ and more, there is a need for optimization algorithms, for which storage requirement and computational

cost per iteration grow not more than linearly in n. An early algorithm of this type is the conjugate gradient (CG) method [5], [4], [15]. It is known that CG worst case convergence rate for quadratic problems is $O(k^{-2})$ (in terms of objective function), where k is the iteration count. This rate of convergence is independent of the problem size and is optimal, *i.e.* it coincides with the complexity of convex smooth unconstrained optimization (see *e.g.* [10]). The extensions of CG to nonlinear functions by Fletcher-Reeves and Polak-Ribière (see *e.g.* [15]) are no longer worst-case optimal.

1.1 Nemirovski-Nesterov methods

The optimality of the quadratic CG method is associated with the following properties:

1. The current gradient is orthogonal to the directions of all previous steps.
2. The current gradient is orthogonal to all previous gradients.
3. The objective function improvement at iteration k is at least $O(\|\mathbf{g}(\mathbf{x}_k)\|^2)$, where $\mathbf{g}(\mathbf{x}_k) = \nabla_{\mathbf{x}} f(\mathbf{x}_k)$ is the gradient of the objective function at iterate $\mathbf{x}_k$.

Nemirovski [9] suggested to relax these requirements, in order to build an optimal method for convex smooth unconstrained optimization:

1. The current gradient should be orthogonal to the sum of all previous steps, *i.e.* orthogonal to $\mathbf{d}_k^1 = \mathbf{x}_k - \mathbf{x}_0$.
2. The current gradient should be orthogonal to a weighted sum of all previous gradients $\mathbf{d}_k^2 = \sum_{i=0}^{k-1} w_i \mathbf{g}(\mathbf{x}_i)$ with pre-specified weights w_i.
3. The optimization at the current iteration should be performed over a subspace, which includes the current gradient $\mathbf{g}(\mathbf{x}_k)$.

These three requirements can be satisfied by a method which sequentially minimizes the objective function over subspaces spanned by the three mentioned vectors: $\mathbf{d}_k^1$, $\mathbf{d}_k^2$, and $\mathbf{g}(\mathbf{x}_k)$. Note that this method is optimal with respect to the number of subspace minimizations, however the overall number of function/gradient evaluations is suboptimal by a factor of $\log k$ [9]. Nemirovski also suggested methods with 2-d and even 1-d subspace optimization instead of 3-d one [12]. Further progress in this direction was achieved by Nesterov [13], [14], [10], who proposed a worst-case optimal algorithm with no line search, which achieves the optimal complexity in terms of function/gradient evaluations.

In practical situations, however (in contrast to the worst case), the mentioned methods often behave even poorer than conventional algorithms like non-linear CG or Truncated Newton (TN) (see *e.g.* [4] for a description of TN). In the current work we present a method, which is equivalent to CG in the quadratic case, and often outperforms CG and TN in non-quadratic case, while preserving worst-case optimality.

1.2 Extended subspace optimization

Our crucial observation is that for many important problems subspace optimization can be implemented extremely fast. This happens, for example, when the objective function is a combination of expensive linear mappings with computationally cheap non-linear functions. It is a typical situation in many applications, like tomography, signal and image processing with Basis Pursuit, pattern recognition with Support Vector Machine, and many others. Another example is constrained optimization, where barrier or penalty aggregate may have this property in linear programming, semidefinite programming, *etc.* In such situations the overall cost of subspace optimization is about one function and gradient evaluation!

Motivated by this observation we tend to increase the dimensionality of the search subspaces and use quite accurate subspace optimization (contrary to the trends of Nemirovski-Nesterov). The first additional vector we include is the last step

$$\mathbf{p}_k = \mathbf{x}_k - \mathbf{x}_{k-1}.$$

There is a deep reason to do so: Iteration of quadratic CG can be defined as an optimization in the subspace of the current gradient $\mathbf{g}(\mathbf{x}_k)$ and the last step $\mathbf{p}_k$ (see *e.g.* [4]). Preserving this property is a natural way to extend CG to the non-quadratic case. Note that Fletcher-Reeves and Polak-Ribière nonlinear CG method lack this property, which could be very helpful: every iteration is guaranteed to be at least as good as steepest descent. On the other hand, by the *expanding manifold* property, quadratic CG achieves minimum over the subspace of the current gradient and all previous steps and gradients. We can approximate this property including several previous steps and gradients into the optimization subspace.

Let us summarize: Using only 2-d subspace optimizations in directions $\mathbf{g}(\mathbf{x}_k)$ and $\mathbf{p}_k$, we get a method, which coincides with CG, when the problem becomes quadratic. This property is favorable in the proximity of the solution, where the problem has a good quadratic approximation. Globally (in our experience) this method behaves better and is more stable then Polak-Ribière CG. Using two additional Nemirovski directions: $\mathbf{d}_k^1 = \mathbf{x}_k - \mathbf{x}_0$ and $\mathbf{d}_k^2 = \sum_{i=0}^{k-1} w_i \mathbf{g}(\mathbf{x}_i)$ with appropriate weights w_i, we guarantee the worst-case optimality of the method. Including more previous steps and gradients into the optimization subspace helps to

further reduce the number of iterations, while moderately increasing the iteration cost.

We also introduce pre-conditioning into this scheme, using pre-multiplication of gradients by an approximate inverse Hessian (in our experiments we have used diagonal approximation). This measure quite often significantly accelerates convergence.

The paper is organized as follows. In Section 2 we describe the sequential subspace optimization algorithm and discuss its properties. In Section 3 we present a way to conduct an efficient minimization of functions in subspace. Section 4 is devoted to computational experiments. Finally, conclusions are summarized in Section 5.

2 Sequential Subspace Optimization (SESOP) algorithm

In this section we describe the SESOP algorithm, which is a general method for smooth unconstrained optimization. We define the algorithm with its various modes, discuss its properties and prove its complexity.

2.1 Construction of subspace structure

In order to define the subspace structure, denote the following sets of directions:

1. **Current gradient**: $\mathbf{g}(\mathbf{x}_k)$ - the gradient at the k'th point $\mathbf{x}_k$.
2. **Nemirovski directions**:
$$\begin{aligned} \mathbf{d}_k^{(1)} &= \mathbf{x}_k - \mathbf{x}_0 \\ \mathbf{d}_k^{(2)} &= \sum_{i=0}^{k} w_i \mathbf{g}(\mathbf{x}_i), \end{aligned} \tag{2}$$
where w_k is defined by
$$w_k = \begin{cases} 1 & \text{for } k = 0 \\ \frac{1}{2} + \sqrt{\frac{1}{4} + w_{k-1}^2} & \text{for } k > 0. \end{cases} \tag{3}$$
3. **Previous directions**:
$$\mathbf{p}_{k-i} = \mathbf{x}_{k-i} - \mathbf{x}_{k-i-1}, \quad i = 0, \ldots, s_1. \tag{4}$$
4. **Previous gradients**:
$$\mathbf{g}_{k-i}, \quad i = 1, \ldots, s_2. \tag{5}$$

The mandatory direction 1 and any subset of directions 2 - 4 can be used to define the subspace structure. We will discuss possible considerations for several constellations.

2.2 Algorithm summary

Let $\mathbf{D}$ be a matrix of the chosen M (column) directions described in Subsection 2.1, and α a column vector of M coefficients. On every iteration we find a new direction $\mathbf{D}\alpha$ in the subspace spanned by the columns of $\mathbf{D}$. The algorithm is summarized as follows:

1. Initialize $\mathbf{x}_k = \mathbf{x}_0$, $\mathbf{D} = \mathbf{D}_0 = \mathbf{g}(\mathbf{x}_0)$.
2. Normalize the columns of $\mathbf{D}$.
3. Find
$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} f\big(\mathbf{x}_k + \mathbf{D}\alpha\big). \tag{6}$$
4. Update current iterate:
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{D}\alpha^*. \tag{7}$$
5. Update matrix $\mathbf{D}$ according to the chosen set of subspace directions in Subsection 2.1.
6. Repeat steps 2 - 5 until convergence.

Implementation notes:

1. The choice of the subspace dimension M, is a trade off between the increase in computational cost per iteration and the possible decrease in number of iterations. Using just 2-d subspace optimizations in directions of the current gradient $\mathbf{g}(\mathbf{x}_k)$ and of the previous step $\mathbf{p}_k$, we get a method, which coincides with CG, when the problem becomes quadratic. This property is favorable in the proximity of the solution, where the problem has a good quadratic approximation. Also globally (in our experience) this method behaves better and is more stable then Polak-Ribière CG.
2. Including two additional Nemirovski directions (2), we guarantee the worst-case optimality of the method (when solving concrete non-worst case classes of problems, these two directions may not bring significant improvement, therefore can be omitted after careful testing). Including more previous steps and gradients into the optimization subspace helps to further reduce the number of iterations, while moderately increasing the iteration cost. The guiding

principle is that the dimension M of the search subspace should not be higher than a few tens or maybe hundreds of directions.

3. For the first $M-1$ iterations, some directions may be a combination of other directions, or may not exist. We take advantage of this fact, to decrease the size of matrix $\mathbf{D}$ for these iterations, and reduce the computation load. After more than M iterations, the size of $\mathbf{D}$ does not change.

4. **Preconditioning** A common practice for optimization speedup is to use a preconditioner matrix. One can use a preconditioned gradient $\mathbf{M}\mathbf{g}(\mathbf{x}_k)$ instead of $\mathbf{g}(\mathbf{x}_k)$, where the matrix $\mathbf{M}$ approximates the inverse of the Hessian at point $\mathbf{x}_k$. There is a trade off between the pre-conditioner calculation time, and the optimization runtime saving.

5. **Newton method in subspace optimization** Basically SESOP is a first order method, which can work using only first order derivatives. In many cases the objective function is twice differentiable, but the Hessian cannot be used due to memory and other limitations. However, in the subspace optimization (step 3 of the algorithm), we often use Newton method because of the small size of this auxiliary problem.

2.3 Complexity analysis

This section is in large extent with help of [11].

Theorem 1 *Let $f(\mathbf{x}) : \mathcal{R}^n \rightarrow \mathcal{R}$ be a smooth convex function, with a Lipschitz continuous gradient $\mathbf{g}(\mathbf{x})$ and Lipschitz constant L, meaning that*

$$\|\mathbf{g}(\mathbf{x}_1) - \mathbf{g}(\mathbf{x}_2)\| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\| \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}^n. \tag{8}$$

Let also each optimization subspace in SESOP be spanned by the current gradient, Nemirovski directions (2), (3)*, and possibly several other vectors, then the worst case complexity is*

$$\epsilon_{N+1} \leq \frac{LR^2}{N^2}, \tag{9}$$

where $\epsilon_{N+1} = f(\mathbf{x}_{N+1}) - f(\mathbf{x}^*)$ is the inaccuracy in objective function at iteration number $N+1$, and R is the radius of the ball around the initial point $\mathbf{x}_0$ where the solution must exist:

$$\|\mathbf{x}^* - \mathbf{x}_0\| \leq R. \tag{10}$$

Proof of Theorem 1 In the following we start with three propositions and then complete the proof.

Proposition 1 *Under the conditions of Theorem 1:*

$$f(\mathbf{x}_{n+1}) \leq f(\mathbf{x}_n) - \frac{\|\mathbf{g}(\mathbf{x}_n)\|^2}{2L}, \tag{11}$$

where n is the iteration index.

Proof of proposition The first and the second directional derivatives of $f(\mathbf{x})$ in the gradient direction $\mathbf{g}(\mathbf{x})$ are

$$f'_{\mathbf{g}}(\mathbf{x}) = \langle \mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x}) \rangle = \|\mathbf{g}(\mathbf{x})\|^2 \tag{12}$$

$$f''_{\mathbf{gg}}(\mathbf{x}) = \mathbf{g}(\mathbf{x})^T \mathbf{H} \mathbf{g}(\mathbf{x}) \leq L\|\mathbf{g}(\mathbf{x})\|^2, \tag{13}$$

where $\mathbf{H}$ is the Hessian matrix at $\mathbf{x}$. Proof of inequality (13) is given in Appendix A.

Consider the minimization of $f(\mathbf{x})$ along the descent direction $-\mathbf{g}(\mathbf{x})$ at some point $\mathbf{x}$. We denote $\varphi(\alpha) \triangleq f(\mathbf{x} - \alpha\mathbf{g})$ and $q(\alpha)$ is a quadratic function which possesses the following properties (see Figure 1):

$$\begin{aligned} q(0) &= \varphi(0) \\ q'(0) &= \varphi'(0) \\ q''(\alpha) &= L\|\mathbf{g}(\mathbf{x})\|^2 \geq \varphi''(\alpha). \end{aligned} \tag{14}$$

It is easy to see that

$$q(\alpha) \geq \varphi(\alpha), \tag{15}$$

therefore the guaranteed decrease $\Delta f(\mathbf{x})$ of the function $f(\mathbf{x})$ at one gradient step with exact line search is higher than the decrease of the corresponding majorant $q(\alpha)$ to its minimum point

$$\Delta f(\mathbf{x}) \geq \frac{q'(0)^2}{2q''}, \tag{16}$$

as illustrated in Figure 1. Using (12), (13) we get

$$\Delta f(\mathbf{x}) \geq \frac{\|\mathbf{g}(\mathbf{x})\|^2}{2L}. \tag{17}$$

Since the gradient is one of SESOP directions, we obtain (11). ■

Proposition 2 *Under the conditions of Theorem 1:*

$$\epsilon_n \leq \langle \mathbf{g}(\mathbf{x}_n), \mathbf{x}_0 - \mathbf{x}^* \rangle. \tag{18}$$

Proof of proposition Denote $\epsilon_n \triangleq f(\mathbf{x}_n) - f(\mathbf{x}^*)$. From (11) we get

$$\epsilon_n - \epsilon_{n+1} \geq \frac{\|\mathbf{g}(\mathbf{x}_n)\|^2}{2L}. \tag{19}$$

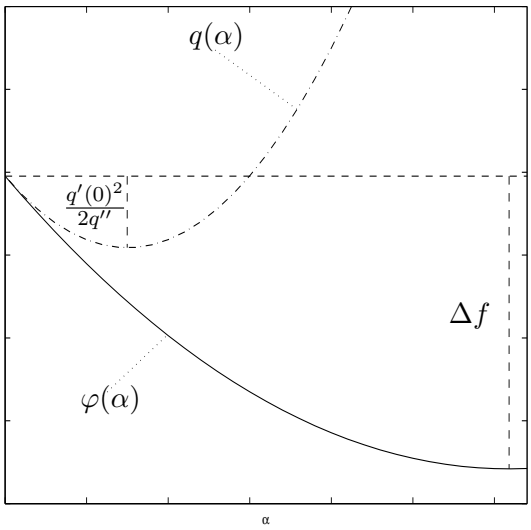

Figure 1: Illustration of equation 16: The function $\varphi(\alpha)$ and its quadratic majorant $q(\alpha)$: $\Delta f(\mathbf{x}) \geq \frac{q'(0)^2}{2q''}$.

Consider the function f along the direction $\mathbf{x} - \mathbf{x}^*$. We denote $\tilde{\varphi}(\alpha) \triangleq f(\mathbf{x} - \alpha(\mathbf{x} - \mathbf{x}^*))$. Due to the convexity of f, $\tilde{\varphi}$ is also convex, meaning that the function is above its linear model

$$\tilde{\varphi}(0) - \tilde{\varphi}(1) \leq \tilde{\varphi}'(0) \cdot 1. \tag{20}$$

Substituting $f(\mathbf{x}) = \tilde{\varphi}(0)$, $f(\mathbf{x}^*) = \tilde{\varphi}(1)$ we get

$$f(\mathbf{x}) - f(\mathbf{x}^*) \leq \langle \mathbf{g}(\mathbf{x}), \mathbf{x} - \mathbf{x}^* \rangle. \tag{21}$$

From (21), the upper bound for the error at iteration n is

$$\epsilon_n \leq \langle \mathbf{g}(\mathbf{x}_n), \mathbf{x}_n - \mathbf{x}^* \rangle. \tag{22}$$

Due to the fact that the objective function has been minimized in the span of $\mathbf{x}_{n-1} - \mathbf{x}_0$ and $\mathbf{x}_n - \mathbf{x}_{n-1}$, it follows that

$$\mathbf{g}(\mathbf{x}_n) \perp \mathbf{x}_n - \mathbf{x}_0. \tag{23}$$

From (22) and (23) we obtain (18). ■

Proposition 3 *Under the conditions of Theorem 1, with weights* w_n*:*

$$\sum_{n=0}^{N} w_n \epsilon_n \leq \sqrt{2L} R \sqrt{\sum_{n=0}^{N} {w_n}^2 (\epsilon_n - \epsilon_{n+1})}. \tag{24}$$

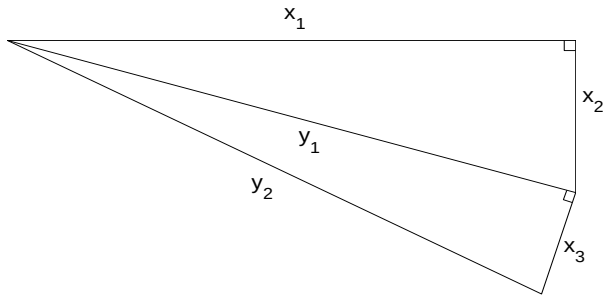

Figure 2: Illustration of equation (28), if $\mathbf{y}_2 = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3$, and every vector is perpendicular to the sum of all previous vectors, then $\|\mathbf{y}_2\|^2 = \|\mathbf{y}_1\|^2 + \|\mathbf{x}_3\|^2 = (\|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2) + \|\mathbf{x}_3\|^2$.

Proof of proposition Lets sum (18) for all steps, with weights w_n:

$$\sum_{n=0}^{N} w_n \epsilon_n \leq \langle \sum_{n=0}^{N} w_n \mathbf{g}(\mathbf{x}_n), \mathbf{x}_0 - \mathbf{x}^* \rangle. \tag{25}$$

Using Cauchy-Schwartz inequality we get

$$\sum_{n=0}^{N} w_n \epsilon_n \leq \| \sum_{n=0}^{N} w_n \mathbf{g}(\mathbf{x}_n) \| \cdot R, \tag{26}$$

where R is given by (10). Since the direction $\sum_{n=0}^{N} w_n \mathbf{g}(\mathbf{x}_n)$ is one of the algorithm subspace directions, then

$$\mathbf{g}(\mathbf{x}_n) \perp \sum_{k=0}^{n-1} w_k \mathbf{g}(\mathbf{x}_k), \tag{27}$$

and following Pythagoras rule (see Figure 2)

$$\| \sum_{n=0}^{N} w_n \mathbf{g}(\mathbf{x}_n) \|^2 = \sum_{n=0}^{N} w_n^2 \|\mathbf{g}(\mathbf{x}_n)\|^2 \leq 2L \sum_{n=0}^{N} w_n^2 (\epsilon_n - \epsilon_{n+1}), \tag{28}$$

where the last inequality is due to (19). Substituting (28) to (26) we obtain (24). ■

By the monotonicity of ϵ_n it follows

$$\epsilon_N \leq \sqrt{2L} R \frac{\sqrt{\sum_{n=0}^{N} {w_n}^2 (\epsilon_n - \epsilon_{n+1})}}{\sum_{n=0}^{N} w_n}. \tag{29}$$

For example, if $w_n = 1 \, \forall n$, we get $\epsilon_N \leq \sqrt{2L} R \epsilon_0 / N$, which is the known complexity of steepest descent [10].

In order to complete the proof of Theorem 1 we show that one can do better using the optimal weights. Rearranging (24) we get

$$\sum_{n=0}^{N} w_n \epsilon_n \leq \sqrt{2L} R \sqrt{w_0^2 \epsilon_0 + (w_1^2 - w_0^2)\epsilon_1 + \cdots + (w_N^2 - w_{N-1}^2)\epsilon_N - w_N^2 \epsilon_{N+1}}. \tag{30}$$

Denote $s \triangleq \sum_{n=0}^{N} w_n \epsilon_n$. In order to evaluate ϵ_N we would like to get

$$s \leq \sqrt{2L} R \sqrt{s - w_N^2 \epsilon_{N+1}}, \tag{31}$$

for this purpose we need to choose weights such that

$$w_n = \begin{cases} w_0^2 & \text{for } n = 0 \\ w_n^2 - w_{n-1}^2 & \text{for } n > 0. \end{cases} \tag{32}$$

The obvious solution is

$$w_n = \begin{cases} 1 & \text{for } n = 0 \\ \frac{1}{2} + \sqrt{\frac{1}{4} + w_{n-1}^2} & \text{for } n > 0, \end{cases} \tag{33}$$

where the last term is the larger root of (32). Notice that for large k, $w_k \approx \frac{k}{2}$. By (31)

$$w_N^2 \epsilon_{N+1} \leq s - \frac{s^2}{2LR^2}. \tag{34}$$

Our goal is the upper bound of ϵ_N so we are interested in the "worst" value of s

$$\hat{s} = \underset{s}{\operatorname{argmax}} \left\{ s - \frac{s^2}{2LR^2} \right\}. \tag{35}$$

The maximum is achieved if $s = LR^2$. Substituting to (34)

$$\epsilon_{N+1} \leq \frac{LR^2}{4w_N^2} \approx \frac{LR^2}{N^2}, \tag{36}$$

which proves Theorem 1. ■

3 Reduced computations for subspace minimization

Consider a function of the form

$$f(\mathbf{x}) = \varphi(\mathbf{A}\mathbf{x}) + \psi(\mathbf{x}). \tag{37}$$

Such functions are very common in many applications. The multiplications $\mathbf{A}\mathbf{x}$ and $\mathbf{A}^T\mathbf{y}$ are usually the most computationally expensive operations for calculating

the function f and its gradient. Our aim is to construct an optimization algorithm which will avoid such operations whenever possible. It is worthwhile emphasizing that we "change the rules" for comparison of computation load between different optimization algorithms. Instead of the common method of counting the number of function and gradient calculations, we will count the number of matrix-vector multiplications. Methods based on subspace optimization often iterate the multi-dimensional minimizer incrementally in the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=1}^{M} \alpha_i \mathbf{r}_i, \tag{38}$$

where the coefficients α_i, the directions $\mathbf{r}_i$ and the number of directions M are determined according to the specific optimization scheme. Such a framework allows us to save a large part of the matrix-vector multiplications originally needed for calculation of the objective function value (37). The term $\mathbf{A}\mathbf{x}_{k+1}$ can be broken into

$$\begin{aligned} \mathbf{A}\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{A}\sum_{i=1}^{M} \alpha_i \mathbf{r}_i \\ &= \mathbf{A}\mathbf{x}_k + \sum_{i=1}^{M} \alpha_i \mathbf{A}\mathbf{r}_i \\ &= \mathbf{v}_0 + \sum_{i=1}^{M} \mathbf{v}_i \alpha_i, \end{aligned} \tag{39}$$

where $\mathbf{v}_i = \mathbf{A}\mathbf{r}_i$. For each new direction $\mathbf{r}_i$ we need to calculate and save one vector ($\mathbf{v}_i$). Total memory requirement is M directions $\mathbf{r}_i$, M matrix-vector multiplications results $\mathbf{v}_i$ and one accumulative vector term $\mathbf{v}_0$. Obviously, as the data dimension n increases, the complexity reduction from using (39) is more significant. For line search operation along a single direction, or subspace minimization along several directions, there is no need to perform any matrix-vector multiplication, since the function and its gradient with respect to α are gained using the pre-calculated set of vectors $\mathbf{v}_i$. For more details on gradient and Hessian calculation in subspace see Appendix C.

4 Computational Experiments

We compared the performance of several algorithms with several large-scale optimization problems: Computational Tomography (CT) and Basis Pursuit (BP). More experiments with Support Vector Machine (SVM) are presented in [8]. The algorithms used were

1. Sequential subspace optimization method, using various numbers of directions;
2. Polak-Ribière nonlinear conjugate gradient method;
3. Truncated-Newton method;
4. Nesterov method [13], [14].

In our experiments the line search methods were cubic interpolation for CG, and back-tracking search with Armijo rule for all other methods, see for example [4]. Newton method was used for subspace minimization in SESOP algorithm. We bring results with and without the usage of diagonal pre-conditioning to show the effect of this practice. The parameters for comparison between algorithms are number of iterations, normalized to two matrix-vector multiplications per iteration, as shown in Table 1, and computation time [1].

Method	**Matrix-vector mult. per iteration**
Subspace	2
Conjugate Gradient	2
Truncated Newton	2 per inner CG iter. + 2 per outer Newton iter.
Nesterov	3

Table 1: Number of heavy operations for different optimization methods.

Notation: SESOPi is an abbreviation for SESOP using i previous directions. When we do not include Nemirovski directions in our scheme, we use the notation SESOPi^-. CG_{fast} means CG with reduced computations for linesearch minimization, as explained in Section 3.

4.1 Computerized Tomography (CT)

Tomography (see *e.g.* [6]) is a method of imaging the inner structure of an object without physically cutting it. Reconstruction is performed from transmission or reflection data collected by illuminating the object from many different directions. Tomography became extremely useful mainly in medical applications as x-ray imaging, emission imaging (PET, SPECT) and ultrasound. In this section we solve the two dimensional straight-ray transmission tomography, in which the object is illuminated by straight rays of high frequency radiation (usually in the x-ray spectrum). Thus, a projection can be treated as line integrals along a beam

[1] Experiments were conducted in MATLAB, on Intel® Xenon™ CPU 2.8GHz core with 2Gb memory, running Linux

of parallel rays. Let $x(u,v)$ be a function supported in $\Omega \in \mathcal{R}^2$, representing the property of the object to be reconstructed. A projection obtained by illuminating the object at angle θ is given by

$$\mathbf{R}(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(u,v)\,\delta(\rho - u\cos\theta - v\sin\theta)\,du\,dv, \tag{40}$$

where δ denotes the Dirac delta function. The function $\mathbf{R}(\rho,\theta)$ is called the Radon transform of x. Recovery of the function x given its Radon transform is done by solving the inverse problem of (40) and is termed reconstruction. Since only a finite number of projections can be acquired (*i.e.* θ is discrete), and a finite number of bins at each projection ($\mathbf{y}$ is discrete), we use the discrete version of the Radon transform, and also discretize the plain (u,v) into "pixels", so the unknown image is represented by a matrix $\mathbf{X} = [x_{ij}]$. For simplicity of presentation we parse the matrix column-wise to a long vector $\mathbf{x}$. The model is

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \xi, \tag{41}$$

where $\mathbf{y}$ is the observation vector, $\mathbf{A}\mathbf{x}$ is the Radon transform, implemented with a projection matrix $\mathbf{A}$, and ξ is Gaussian noise. The structure of the projection matrix $\mathbf{A}$ is described in Appendix B. In emission or transmission tomography the noise is Poisson, so this is a simplified example only to demonstrate the performance of the optimization.

Reconstruction of sparse images Suppose that the original image is known to be sparse (say, constructed from "wires"). This may be important for example in recovering blood vessels filled by a contrast material. A convenient way to enforce sparsity of a solution, while preserving convexity of the objective function is to add some l_1-norm penalty term (see *e.g.* [1]). This brings us to the following penalized least square formulation

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \mu\|\mathbf{x}\|_1, \tag{42}$$

where $\mathbf{y}$ is the observed noisy projection data and μ is a regularization parameter. In order to use smooth optimization, we approximate the l_1-norm by a smooth function $\mathbf{1}^T\psi(\mathbf{x})$ (for details see Appendix D).

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \mu\mathbf{1}^T\psi(\mathbf{x}). \tag{43}$$

Some ways to determine the value for the regularization parameter μ are suggested in [1] and require a crude estimation of the noise variance.

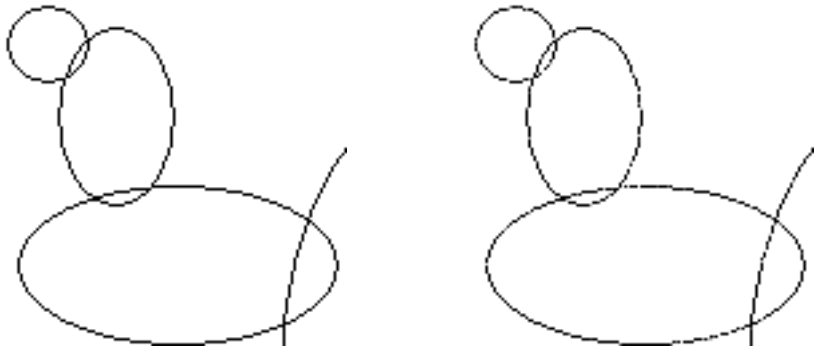

Figure 3: Sparse tomography: Left: Original sparse image, Right: Reconstructed image

Numerical Results In our experiment we created sparse images of ellipses of sizes $128^2, 256^2$ pixels (see Figure 3), and used 100 uniformly spaced angles of projections, with Gaussian noise (std = $0.08 \times$ (image maximum range)). We compared the time and number of iterations of different methods to solve (43) to an accurate solution ($\|\nabla f\| \leq 10^{-4}$) with various optimization algorithms, and the convergence to 'good result' solution. 'Good result' solution was defined when the PSNR first reached 0.01dB from the final stable PSNR. Figures 4,5 present the inaccuracy in objective function and the PSNR as a function of iteration number. Iteration numbers and runtime are summarized in Table 2. In all cases, the subspace method outperforms the CG method by 25% less iterations to full convergence as well as to 'good results'. Runtime of subspace method is better than CG with reduced linesearch computations (CG_{fast}). The number of matrix-vector multiplication decreases as the number of previous directions used by SESOP method becomes higher, but overall runtime with iterations may increase. The best choice in this example is to use a single previous direction and to include Nemirovski directions (SESOP1). Pre-conditioning improved convergence almost by a factor of 2.

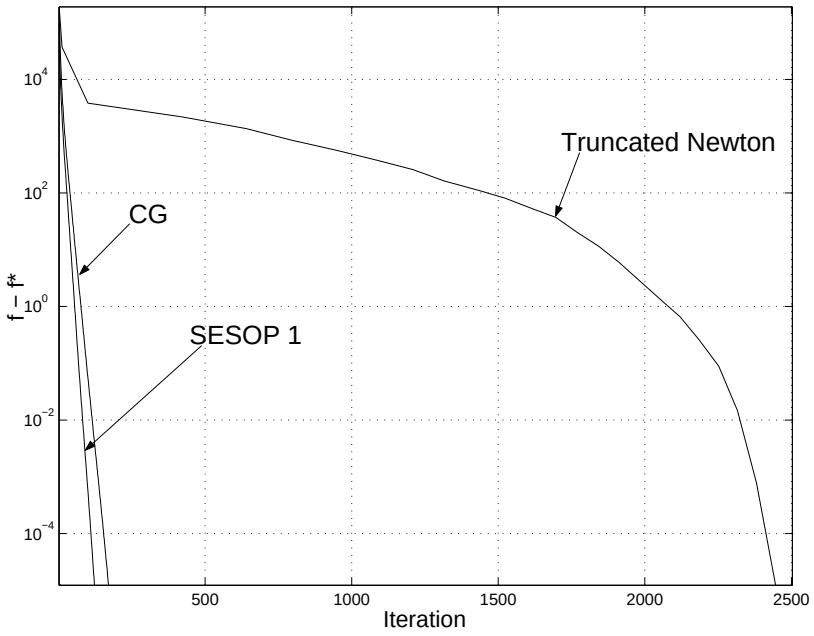

Figure 4: Sparse tomography: Inaccuracy in objective function [log scale] with iterations

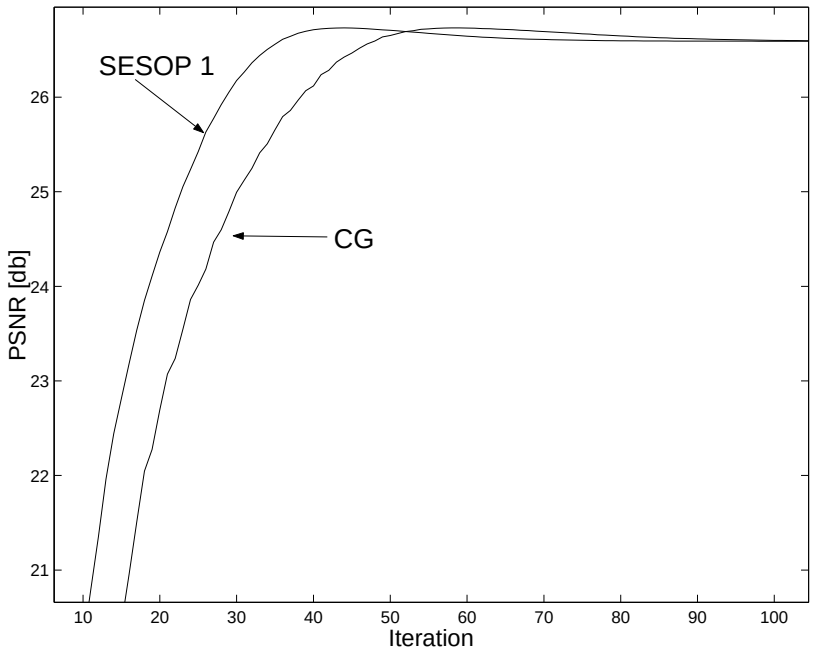

Figure 5: Sparse tomography: PSNR with iterations

		No pre-conditioning				**Pre-conditioning**			
Image		**Convergence**		**Good results**		**Convergence**		**Good results**	
size	**Method**	**iter**	**time**	**iter**	**time**	**iter**	**time**	**iter**	**time**
128^2	Nesterov	∞	∞	159	32.5				
	SESOP0	3335	638	67	14.5	1734	334	190	38.4
	SESOP1$^-$	478	85.9	70	14	194	36	46	9.66
	SESOP1	349	72	70	15.5	138	28.4	36	8.21
	SESOP8	314	91	49	15.7	132	38.1	35	10.9
	SESOP32	270	171	43	21.6	105	60	30	11.4
	SESOP128	219	334	42	19.7	87	53.8	30	11.4
	CG$_{fast}$	467	150	70	17	294	118.6	48	13
	CG	465	1148	70	68.7	294	1245	48	66.4
	TN	3821	511	2637	372	2632	355	2053	280
256^2	Nesterov	∞	∞	225	210				
	SESOP0	∞	∞	89	98	2644	2225	245	216
	SESOP1$^-$	720	620	96	86.5	256	209	54	50.3
	SESOP1	528	507	74	69	182	166	41	39.4
	SESOP8	490	631	70	86	177	199	42	52
	SESOP32	400	860	59	118	142	286	35	55
	SESOP128	322	2300	54	112	109	290	35	54.2
	CG$_{fast}$	700	936	97	102.4	377	627	54	63.2
	CG	705	8998	97	431	377	7983	54	358
	TN	6876	4300	3065	1911	5050	3208	2729	1728

Table 2: Sparse tomography: Iterations and CPU runtime [sec] to convergence ($\|\nabla f\| \leq 10^{-4}$), and to 'good results' (PSNR reached 0.01dB from the final PSNR). ∞ - no convergence in 5000 iterations.

4.2 Basis Pursuit (BP)

Basis Pursuit [1] is a way for decomposing a signal into a sparse superposition of dictionary elements, using l_1-norm penalization. When the signal is sparse enough, it is equivalent to l_0 minimization, where $l_0(\alpha)$ stands for the number of non-zero elements in α [3]. BP in highly overcomplete dictionaries leads to large-scale optimization problems. We bring an example of image de-noising with contourlets dictionary [2], [7]. The noisy picture $\mathbf{y}$ is described as

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{44}$$

where $\mathbf{x}$ is the original picture (parsed column-wise), and $\mathbf{n}$ is Gaussian noise with variance σ_n^2. We assume $\mathbf{x} = \mathbf{\Phi}\alpha$ where $\mathbf{\Phi}$ is a "synthesis" operator (equivalent to a matrix of basis functions in its columns). Assuming Laplace distribution of the original picture's coefficients:

$$p(\alpha) \sim \prod_i e^{-\frac{\sqrt{2}}{\sigma_i}|\alpha_i|}, \tag{45}$$

the MAP estimation for the coefficients is achieved by maximizing the log-likelihood

$$\min_{\alpha} \frac{1}{2\sigma_n^2}\|\mathbf{\Phi}\alpha - \mathbf{y}\|_2^2 + \sum_i \lambda_i|\alpha_i|. \tag{46}$$

In order to use smooth optimization, we approximate the absolute value by a smooth function (see Appendix D). The free parameters $\lambda_i = \frac{\sqrt{2}}{\sigma_i}$ can be estimated, for example, by averaging coefficients of the noisy picture in the contourlets domain neighborhood [7].

Numerical Results The experiment was conducted on the popular image 'peppers'. For picture size of 256^2 pixels, the number of coefficients is $87,296$. We compared the time and number of iterations of different methods to solve (46) to an accurate solution ($\|\nabla f\| \leq 10^{-4}$) with various optimization algorithms, and the convergence to 'good results'. 'Good results' were defined when the PSNR first reached $0.01dB$ from the final stable PSNR. Inaccuracy in objective function is shown in Figure 6, Figure 7 presents the PSNR measure, and image results are shown in Figure 8. Iteration numbers and runtime are summarized in Table 3. Pre-conditioning improved convergence significantly, by factors of $10^2 - 10^3$. The number of previous directions in the subspace method had very little impact on the results when pre-conditioning was used, otherwise 8 or 32 previous directions were usually the best choice. The effect of omitting Nemirovski directions (SESOP1$^-$) is not very dramatic. The most important result in this example is that the subspace method converged faster than CG with reduced linesearch computations (CG_{fast}), although for identical number of iterations for 'Good results' the CG_{fast} was faster (pictures of sizes 256^2, 512^2).

4.3 Other Numerical Experiments with SESOP

In [8] we bring more numerical results with SESOP in the area of pattern recognition using Support Vector Machines. Just to summarize briefly: we have solved six problems with $10^3 - 10^6$ variables. SESOP was consistently faster than Nesterov method, CG and TN, on average outperforming TN ten times and CG about two times.

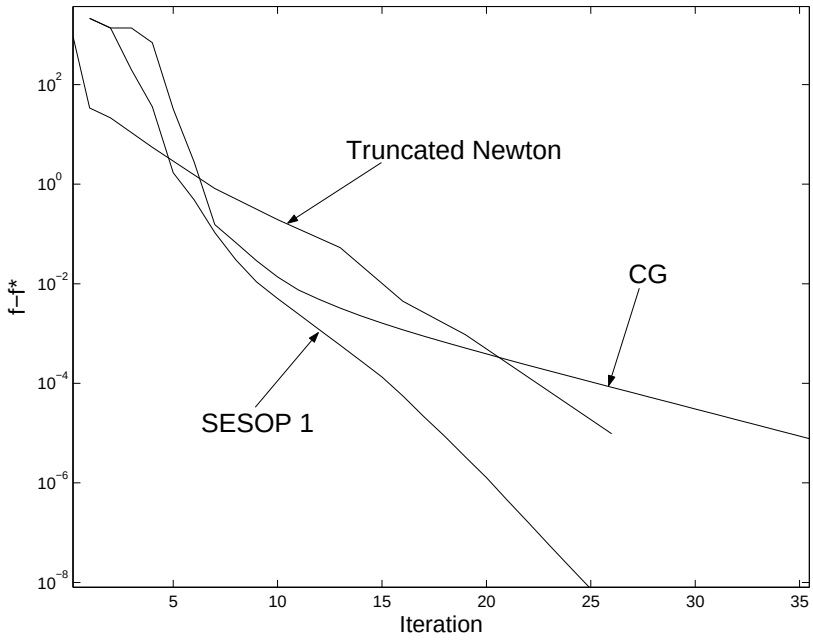

Figure 6: Basis pursuit: Inaccuracy in objective function [log scale] with iterations

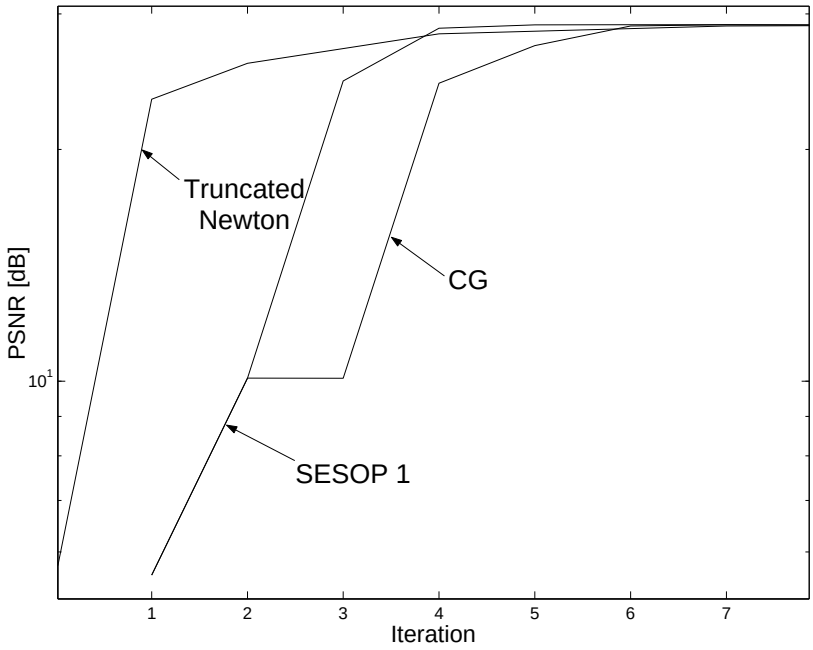

Figure 7: Basis pursuit: PSNR [dB] with iterations

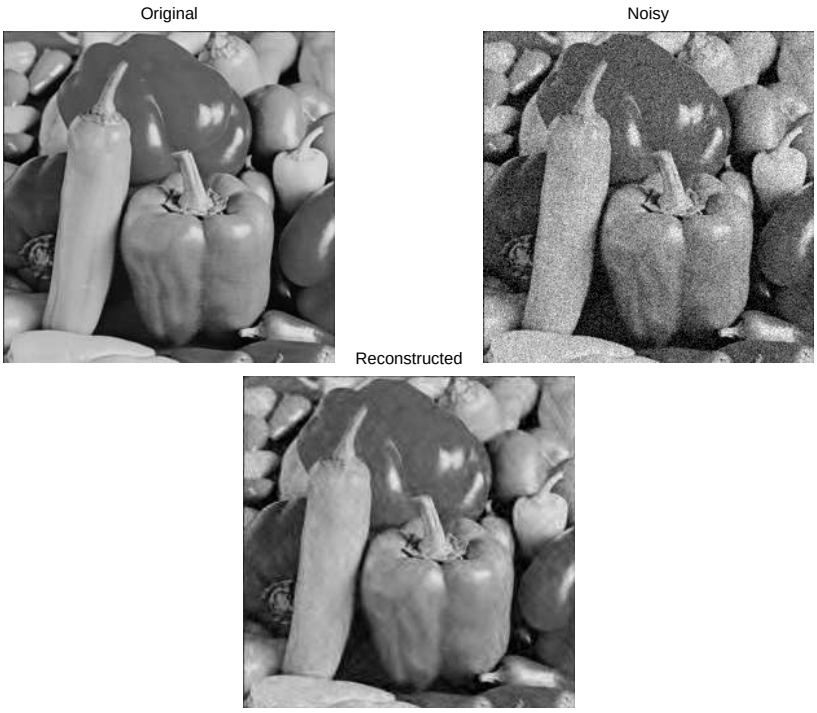

Figure 8: Basis pursuit de-noising example for picture 'peppers': Top left - original image, Top right - noisy image, Bottom - reconstructed image (PSNR increase 8dB)

		No pre-conditioning				**Pre-conditioning**			
Image		**Convergence**		**Good results**		**Convergence**		**Good results**	
size	**Method**	**iter**	**time**	**iter**	**time**	**iter**	**time**	**iter**	**time**
128^2	SESOP1$^-$	∞	∞	1275	218.21	25	5.39	5	1.23
	SESOP1	∞	∞	957	173.17	25	5.42	5	1.09
	SESOP2	∞	∞	1135	211.48	25	5.54	5	1.06
	SESOP8	∞	∞	842	208.43	25	6.47	5	1.05
	SESOP32	4860	2363.61	410	228.02	25	6.68	5	1.08
	CG$_{fast}$	∞	∞	1186	732.41	55	8.22	7	1.28
	CG	∞	∞	1186	189.51	56	11.42	7	4.67
	TN	12655	922.02	4	461.4	56	7.88	12	5.52
256^2	SESOP1$^-$	∞	∞	∞	∞	83	66.77	8	8.05
	SESOP1	∞	∞	∞	∞	83	69.62	8	8.07
	SESOP2	∞	∞	3900	3200.07	83	74.36	8	8.44
	SESOP8	∞	∞	3400	3532.76	80	92.67	8	8.63
	SESOP32	∞	∞	2000	4704.36	79	154.17	8	8.72
	CG$_{fast}$	∞	∞	∞	∞	335	211	8	7.18
	CG	∞	∞	∞	∞	339	211.73	8	25.97
	TN	∞	∞	9581	3272.76	306	126.99	3	3.68
512^2	SESOP1$^-$	∞	∞	∞	∞	75	349.22	8	48.2
	SESOP1	∞	∞	∞	∞	74	366.75	8	52.06
	SESOP2	∞	∞	∞	∞	75	382.23	8	51.7
	SESOP8	∞	∞	∞	∞	74	450.54	8	53.01
	SS32	∞	∞	∞	∞	74	681.62	8	50.94
	CG$_{fast}$	∞	∞	∞	∞	268	984.15	10	45.13
	CG	∞	∞	∞	∞	277	1499.22	10	245.4
	TN	∞	∞	∞	∞	156	432.18	9	184.15

Table 3: Basis Pursuit de-noising example: Iterations and CPU runtime [sec] to convergence, and to 'good results'. ∞ - no convergence in 5000 iterations.

5 Conclusions

We have demonstrated that SESOP is an efficient tool for large-scale unconstrained optimization. The main advantages of SESOP are optimal worst-case complexity for smooth convex unconstrained problems, low memory requirements and low computation load per iteration. Unconstrained optimization is a building block for many constrained optimization techniques, which makes SESOP a promising candidate for embedding into many existing solvers.

Acknowledgment. We are grateful to Arkadi Nemirovski for his most useful advice and support. Our thanks to Boaz Matalon for the Contourlet code and explanations, and to Dori Peleg for the references to pattern recognition data sets. This research has been supported in parts by the "Dvorah" fund of the Technion and by the HASSIP Research Network Program HPRN-CT-2002-00285, sponsored by the European Commission. The research was carried out in the Ollendorff Minerva Center, funded through the BMBF.

Appendix

A Proof of (13)

Consider the first order Taylor expansion of $\nabla_{\mathbf{x}} f(\mathbf{x})$ around $\mathbf{x}_0$

$$\nabla_{\mathbf{x}} f(\mathbf{x}_0 + t \cdot \mathbf{v}) = \nabla_{\mathbf{x}} f(\mathbf{x}_0) + \nabla^2_{\mathbf{x}} f(\mathbf{x}_0)^T t\mathbf{v} + o(t\mathbf{v}). \tag{47}$$

After a simple manipulation and division by t we get

$$\frac{\nabla_{\mathbf{x}} f(\mathbf{x}_0 + t \cdot \mathbf{v}) - \nabla_{\mathbf{x}} f(\mathbf{x}_0)}{t} = \frac{\nabla^2_{\mathbf{x}} f(\mathbf{x}_0)^T t\mathbf{v} + o(t\mathbf{v})}{t}. \tag{48}$$

The value of the right hand side, when $t \to 0$, is

$$\lim_{t\to 0} \frac{\nabla^2_{\mathbf{x}} f(\mathbf{x}_0)^T t\mathbf{v} + o(t\mathbf{v})}{t} = \nabla^2_{\mathbf{x}} f(\mathbf{x}_0)\mathbf{v}. \tag{49}$$

Explanation:

$$\begin{aligned} \lim_{t\to 0} \left| \frac{o(t\mathbf{v})}{t} \right| &= \|\mathbf{v}\| \lim_{t\to 0} \frac{\|o(t\mathbf{v})\|}{|t|\,\|\mathbf{v}\|} \\ &= \|\mathbf{v}\| \lim_{t\to 0} \frac{\|o(t\mathbf{v})\|}{\|t\,\mathbf{v}\|} \\ &= \|\mathbf{v}\| 0 = 0. \end{aligned} \tag{50}$$

Using (8), the norm of (48) is bounded by:

$$\left\| \frac{\nabla_{\mathbf{x}} f(\mathbf{x}_0 + t \cdot \mathbf{v}) - \nabla_{\mathbf{x}} f(\mathbf{x}_0)}{t} \right\| \leq L\|\mathbf{v}\|. \tag{51}$$

Since this equality holds for every t, we can take $t \to 0$

$$\|\nabla^2_{\mathbf{x}} f(\mathbf{x}_0)\mathbf{v}\| \leq L\|\mathbf{v}\| \quad \forall \mathbf{x}_0 \in \mathcal{R}^n. \tag{52}$$

From Cauchy-Schwartz inequality

$$\|\mathbf{v}^T \nabla^2_{\mathbf{x}} f(\mathbf{x}_0)\mathbf{v}\| \leq \|\mathbf{v}^T\| \cdot \|\nabla^2_{\mathbf{x}} f(\mathbf{x}_0)\mathbf{v}\|. \tag{53}$$

Substituting (53) into (52), with $\mathbf{v} = \nabla_{\mathbf{x}} \mathbf{f}(\mathbf{x})$ proves (13). $\square$

B Radon Projection Matrix

Denote $\mathcal{R}_\theta\{\mathbf{x}\}$ the Radon transform along direction θ, where $\mathbf{x} \in \mathcal{R}^N$ is the image $\mathbf{X}$ parsed into a long vector, column-wise. Using the linear property of the transform we define $\mathbf{A}_\theta$ the Radon projection matrix in direction θ in the following way

$$\begin{aligned} \mathcal{R}_\theta\{\mathbf{x}\} &= \mathcal{R}_\theta\{\sum_{i=1}^{N} \mathbf{e}_i x_i\} \\ &= \sum_{i=1}^{N} \mathcal{R}_\theta\{\mathbf{e}_i\} x_i \\ &= \begin{pmatrix} | & | & & | \\ \mathcal{R}_\theta\{\mathbf{e}_1\} & \mathcal{R}_\theta\{\mathbf{e}_2\} & \cdots & \mathcal{R}_\theta\{\mathbf{e}_N\} \\ | & | & & | \end{pmatrix} \mathbf{x} \\ &= \mathbf{A}_\theta \mathbf{x}, \end{aligned} \tag{54}$$

where $\mathbf{e}_i$ is a vector of zeros except for 1 in its i'th element. The Radon projection matrix using a set of angles θ_i, $i = 0 \ldots M$ is defined

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{\theta_0} \\ \vdots \\ \mathbf{A}_{\theta_M} \end{pmatrix}. \tag{55}$$

C Gradient and Hessian calculation in subspace

Consider an objective function of the form (37). For simplicity of the analysis we denote

- K_φ - Number of mathematical operations per vector element for calculating $\varphi(\mathbf{u})$ or $\nabla_{\mathbf{u}}\varphi(\mathbf{u})$.

- K_ψ - Number of mathematical operations per vector element for calculating $\psi(\mathbf{u})$ or $\nabla_{\mathbf{u}}\psi(\mathbf{u})$.

Each outer iteration of SESOP requires a single evaluation of the objective gradient with respect to $\mathbf{x}$ ($mK_\varphi + nm + nK_\psi$ operations), and an update of the directions matrix: one matrix-gradient multiplication, one weighted gradient summation to Nemirovski's direction, and up to two vector normalization operations (additional $nm + 3n$ operations)[2].

At the inner iteration of the subspace optimization (6), the basic calculations are usually function and gradient evaluation with respect to α, neither of which requires a matrix-vector multiplication, as explained in Section 3. Function calculation requires $m(M+1) + mK_\varphi + n(M+1) + nK_\psi$ operations, and the gradient with respect to α requires $mM + mK_\varphi + nK_\psi$ additional operations.

When Newton method is used for the subspace optimization, calculation of the Hessian with respect to α will be required as well. Assuming the number of operations for second order derivative calculation is similar to the number of operations for function calculation, the Hessian with respect to α requires additional $mK_\varphi + mM^2 + nK_\psi + nM^2$ operations.

D Absolute value smoothing techniques

In our experiments we considered the following smooth functions to approximate absolute value

$$\begin{aligned} \psi_1(s) &= \sqrt{s^2+\epsilon^2} \\ \psi_2(s) &= \frac{\epsilon|s| - \log\left(\epsilon|s|+1\right)}{\epsilon} \\ \psi_3(s) &= \epsilon\left(\left|\frac{t}{\epsilon}\right| + \frac{1}{|\frac{t}{\epsilon}|+1} - 1\right), \end{aligned} \tag{56}$$

where ϵ is a positive smoothing parameter. The approximations become accurate when $\epsilon \to 0$. Function ψ_3 was found to be substantially faster than the other two functions in our Matlab implementation. We use the notation $\psi(\mathbf{s}) = (\psi(s_1)\dots\psi(s_N))^T$.

[2] When the matrix $\mathbf{A}$ is significantly sparse, the number of operations for matrix-vector multiplications is reduced accordingly.

Sequential update of the smoothing parameter Whenever a very small value of the smoothing parameter is required for good performance of the problem solution, the direct unconstrained optimization may become difficult. In this situation one can use a sequential nested optimization: Starting with a moderate value of ϵ, optimize the objective function to a reasonable accuracy, then reduce ϵ by some factor and perform the optimization again, starting from the currently available solution, and so on... Another alternative is to use the smoothing method of multipliers [17], [18], which combines the ideas of Lagrange multipliers with the ideas of smoothing of non-smooth functions, and provides a very accurate solution.

References

[1] S. S. CHEN, D. L. DONOHO, AND M. A. SAUNDERS, *Atomic decomposition by basis pursuit*, SIAM Journal on Scientific Computing, 20 (1999), pp. 33–61.

[2] M. N. DO AND M. VETTERLI, *The contourlet transform: An efficient directional multiresolution image representation*, IEEE Transactions on Image Processing, to appear, (2005).

[3] M. ELAD AND M. ZIBULEVSKY, *A probabilistic study of the average performance of the basis pursuit*, submitted to IEEE Trans. On Information Theory, (2004).

[4] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Practical Optimization*, Academic Press, 1981.

[5] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409–436.

[6] A. C. KAK AND M. SLANEY, *Principles of Computerized Tomographic Imaging*, Society of Industrial and Applied Mathematics, 2001.

[7] B. MATALON, M. ZIBULEVSKY, AND M. ELAD, *Improved denoising of images using modeling of a redundant contourlet transform*, in SPIE, Optics and Photonics, Wavelets XI, to be published, 2005.

[8] G. NARKISS AND M. ZIBULEVSKY, *Support vector machine via sequential subspace optimization*, technical report CCIT No. 557, Technion - The Israel Institute of Technology, faculty of Electrical Engineering, Haifa, Israel, 2005.

[9] A. NEMIROVSKI, *Orth-method for smooth convex optimization (in russian).*, Izvestia AN SSSR, Ser. Tekhnicheskaya Kibernetika (the journal is translated to English as Engineering Cybernetics. Soviet J. Computer & Systems Sci.), 2 (1982).

[10] ——, *Efficient methods in convex programming*, Technion - The Israel Institute of Technology, faculty of Industrial Engineering and Managments, 1994.

[11] ——, *Personal communications*, 2005.

[12] A. NEMIROVSKI AND D. YUDIN, *Information-based complexity of mathematical programming (in russian).*, Izvestia AN SSSR, Ser. Tekhnicheskaya Kibernetika (the journal is translated to English as Engineering Cybernetics. Soviet J. Computer & Systems Sci.), 1 (1983).

[13] Y. NESTEROV, *A method for unconstrained convex minimization problem with the rate of convergence o(1/n^2) (in russian)*, Doklady AN SSSR (the journal is translated to English as Soviet Math. Docl.), 269 (1983), pp. 543–547.

[14] ——, *Smooth minimization of non-smooth functions*, CORE discussion paper, Universitè catholique de Louvain, Belgium, 2003.

[15] J. SHEWCHUK, *An introduction to the conjugate gradient method without*

the agonizing pain, technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994.

[16] V. VAPNIK, *Statistical learning theory*, Wiley-Interscience, New York, 1998.

[17] M. ZIBULEVSKY, *Penalty/Barrier Multiplier Methods for Large-Scale Nonlinear and Semidefinite Programming*, PhD thesis, Technion – Israel Institute of Technology, 1996. http://ie.technion.ac.il/˜mcib/.

[18] M. ZIBULEVSKY, *Smoothing method of multipliers for sum-max problems*, tech. report, Dept. of Elec. Eng., Technion, 2003. http://ie.technion.ac.il/˜mcib/.