TECHNICAL UNIVERSITY OF CRETE
ELECTRICAL & COMPUTER ENGINEERING SCHOOL

Course Exercises

Student name: *Georgios Apostolakis (2020039028)*

Course: *Advanced Topics in Convex Optimization* – Professor: *Athanasios P. Liavas*
Semester: *Spring, 2021-2022*

Exercise 1

As unit simplex is defined the subset of $\mathbb{R}^n$ such that:

$$\Delta_n = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{e}^T \mathbf{x} = 1, \mathbf{x} \geq \mathbf{0} \right\}$$

where $\mathbf{e}$ is the unit vector $[1, \ldots, 1]^T \in \mathbb{R}^n$.

For any $\mathbf{x} \in \mathbb{R}^n$, its projection $\mathbf{y} = \mathbf{P}_{\Delta_n}(\mathbf{x}) \in \mathbb{R}^n$ onto the unit simplex is the (unique) solution of the following optimization problem:

$$\begin{aligned} \underset{\mathbf{y} \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \\ \text{subject to} \quad & \mathbf{y} \geq \mathbf{0}, \\ & \mathbf{e}^T \mathbf{y} = 1 \end{aligned} \tag{1}$$

Moreover, according to Corrolary 6.29 in [1], $\mathbf{P}_{\Delta_n}(\mathbf{x})$ is given by:

$$\mathbf{P}_{\Delta_n}(\mathbf{x}) = [\mathbf{x} - \mu_* \mathbf{e}]_+ \text{, where } \mu_* \text{ is a solution of } \mathbf{e}^T [\mathbf{x} - \mu_* \mathbf{e}]_+ - 1 = 0. \tag{2}$$

The solution μ_* of the above equation can be approximated efficiently by a variety of methods, such as the bisection procedure. Moreover, it's worth noting that $[\mathbf{x}]_+ = \max_{\text{elementwise}}(\mathbf{x}, \mathbf{0})$.

We implement the above problem in MATLAB, for a random point $\mathbf{x} \in \mathbb{R}^2$. Initially, we compute projection $\mathbf{P}_{\Delta_n}(\mathbf{x})$ from equation (2), and afterwards we solve the optimization problem (1) in order to check the precision of the solution with the CVX software. The projections returned by the 2 methods are given below, and it can be seen that they are extremely close to each other.

$$\mathbf{P}_{\Delta_n}^{EQ.2}(\mathbf{x}) = \begin{bmatrix} 0.4544659 \\ 0.5455341 \end{bmatrix}, \qquad \mathbf{P}_{\Delta_n}^{CVX}(\mathbf{x}) = \begin{bmatrix} 0.4544622 \\ 0.5455378 \end{bmatrix}$$

Moreover, in figure 1 we can see graphically the point $\mathbf{x}$, the unit simplex, as well as the projection of $\mathbf{x}$ onto the unit simplex. As described above, the projection is the point of the unit simplex that minimizes the distance from $\mathbf{x}$ to it.

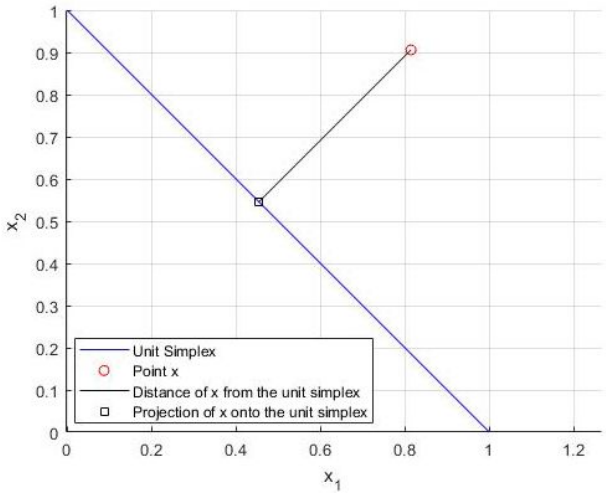


Figure 1: Projection of a random point **x** onto the unit simplex

Finally, the MATLAB code which produces the above results can be found in file `convexopt_ex01.m`.

Exercise 2

We are given 2 sets S_1, S_2 and we have to compute a point of their intersection. This task can be easily achieved by finding the projection of a random point $\mathbf{x} \in \mathbb{R}^n$ onto their intersection $S_1 \cap S_2$. Initially, let's assume that $\mathbf{A} \in \mathbb{R}^{mxn}$ and $\mathbf{b} \in \mathbb{R}^m$. Both sets are non-empty, closed and convex. Moreover, we already know from Lemma 6.26 in [1], that the projections of a random point $\mathbf{x}$ onto S_1 and S_2 are equal to:

$$\mathbf{P}_{S_1}(\mathbf{x}) = \mathbf{x} - \mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T\right)^{-1} (\mathbf{A}\mathbf{x} - \mathbf{b}), \qquad \mathbf{P}_{S_2}(\mathbf{x}) = [\mathbf{x}]_+. \tag{3}$$

Consequently, we may use either the alternating projection algorithm, or the greedy projection algorithm, both of which are presented at section 8.2.3 of [1]. An important assumption which has to be made (and will be verified at the beginning of the implementation) is that $S_1 \cap S_2 \neq \emptyset$, otherwise convergence will never happen.

- The alternating projection method consists of the iterative execution of the following update, starting from any $\mathbf{x}_0 \in S_2$:

$$\mathbf{x}_{k+1} = \mathbf{P}_{S_2}\left(\mathbf{P}_{S_1}\left(\mathbf{x}_k\right)\right) \tag{4}$$

  and terminates when the iterations converge.

  By combining equations (3) and (4), we get the update equation for the given sets S_1, S_2:

$$\mathbf{x}_{k+1} = \left[\mathbf{x}_k - \mathbf{A}^T \left(\mathbf{A}\mathbf{A}^T\right)^{-1} (\mathbf{A}\mathbf{x}_k - \mathbf{b})\right]_+ \tag{5}$$

- When the dimensions of the matrices in equation (5) are large, then the update may be difficult to be computed. Thus, the computations have to be made simpler, which can happen by utilizing the greedy algorithm. More specifically, let's denote with $\mathbf{a}_i^T$ the i-th row of matrix $\mathbf{A}$. Also, let's define:

$$T_i = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}_i^T \mathbf{x} = b_i\right\}, i = 1, 2, \ldots, m, \quad T_{m+1} = S_2 = \mathbb{R}^n_+$$

  Notice that the projection of $\mathbf{x}$ onto $S_1 \cap S_2$ is the same with the projection of $\mathbf{x}$ onto $\cap_{i=1}^{m+1} T_i$. Moreover, from Lemma 6.26 in [1] we know that:

$$\mathbf{P}_{T_i}(\mathbf{x}) = \mathbf{x} - \frac{\mathbf{a}_i^T \mathbf{x} - b_i}{\|\mathbf{a}_i\|_2^2} \mathbf{a}_i, \ \forall i = 1, \ldots, m \tag{6}$$

  The greedy projection method for m sets $T_1, T_2, \ldots, T_m$ consists of the iterative execution of the following update, starting from any random point $\mathbf{x}_0$:

$$\mathbf{x}_{k+1} = \mathbf{P}_{T_{i_k}}\left(\mathbf{x}_k\right), \text{ where } i_k \in \operatorname{argmax}_{j=1,2,\ldots,m} \left\|\mathbf{x}_k - \mathbf{P}_{S_i}\right\| \tag{7}$$

  and terminates when the iterations converge.

  By combining equations (3), (6) and (7), we get the update equation of the greedy algorithm for the $m+1$ sets $T_1, T_2, \ldots, T_{m+1}$:

- $i_k \leftarrow \text{argmax}_{j=1,\ldots,m} \frac{\left|\mathbf{a}_j^T \mathbf{x}_k - b_j\right|}{\|\mathbf{a}_j\|_2}$
- $\mathbf{x}_{k+1} = \max\left\{[\mathbf{x}_k]_+, \frac{\left|\mathbf{a}_{i_k}^T \mathbf{x}_k - b_{i_k}\right|}{\|\mathbf{a}_{i_k}\|_2}\right\}$

Finally, we define for both algorithms the function $f_v(\mathbf{x}_k)$, which indicates how much the constraints are violated at iteration k. More specifically, $f(\mathbf{x}_k)$ indicates the maximum of the distances of $\mathbf{x}_k$ from the sets T_i. When $f(\mathbf{x}_k)$ is less than a threshold ϵ, then it can be assumed that the algorithm has converged.

$$f_v(\mathbf{x}_k) = \max_{j=1,\ldots,m+1}\left\{\left\|\mathbf{x}_k - \mathbf{P}_{T_j}(\mathbf{x}_k)\right\|_2\right\} = \max\left\{\max_{j=1,\ldots,m}\left\{\frac{\left|\mathbf{a}_j^T \mathbf{x}_k - b_j\right|}{\|\mathbf{a}_j\|_2}\right\}, \|\mathbf{x}_k - [\mathbf{x}]_+\|_2\right\}$$

In figure 2 we can see the value of $f_v(\mathbf{x}_k)$ while the iterations progress, for an implementation of both algorithms. Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn randomly. In what regards their dimensions, we have set $n = 8$ and $m = 4$ ($\mathbf{A} \in \mathbb{R}^{4x8}$, $\mathbf{x} \in \mathbb{R}^8$, $\mathbf{b} \in \mathbb{R}^4$). Moreover, before executing the algorithms we checked with CVX software that the intersection of S_1 with S_2 is a non-empty set.

It can be easily noticed that the alternating projection algorithm is significantly faster and reaches a better accuracy than the greedy projection method. On the other hand, the greedy algorithm has to perform much simpler computations than the alternating projection algorithm, especially for high-dimensional problems.

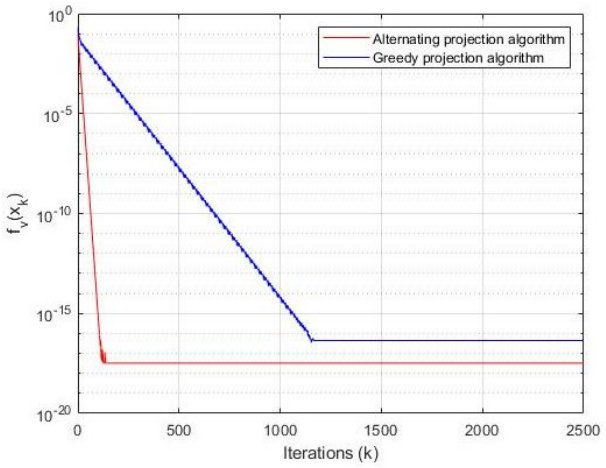


Figure 2: Comparison of projection algorithms for dimensions $n = 8$ & $m = 4$.

Finally, the MATLAB code which produces the above results can be found in file `convexopt_ex02.m`.

Exercise 3

Let $\Delta_n = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0}, \sum_{i=1}^n x_i = 1\}$. On this exercise we have to solve the following optimization problem:

$$\underset{\mathbf{x} \in \Delta_n}{\text{minimize}} \quad \mathbf{c}^T\mathbf{x}$$

The above optimization problem is equivalent with the following one (set $\mathbf{e} = \mathbf{1}_n$):

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & f_0(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & f_i(\mathbf{x}) = -x_i \leq 0, \quad i = 1, \ldots, n, \\ & h_1(\mathbf{x}) = \mathbf{e}^T\mathbf{x} - 1 = 0 \end{aligned} \tag{8}$$

Let $\mathbf{x}_* \in \mathbb{R}^n$ be the optimal point, $\mathbf{u} \in \mathbb{R}^n$ and $v \in \mathbb{R}$. Moreover, the Slater condition is satisfied (choose any $\mathbf{x} > 0$, then it is true that $f_i(\mathbf{x}) < 0$). The KKT conditions are:

$$\nabla f_0(\mathbf{x}_*) + \sum_{i=1}^n u_i \nabla f_i(\mathbf{x}_*) + \mathbf{e}v = \mathbf{0} \Leftrightarrow \mathbf{c} - \mathbf{u} + \mathbf{e}v = \mathbf{0} \overset{\forall i}{\Leftrightarrow} c_i - u_i + v = 0 \tag{9}$$

$$\forall i = 1, \ldots, n, \quad u_i \geq 0 \tag{10}$$

$$\forall i = 1, \ldots, n, \quad u_i f_i(\mathbf{x}_*) = 0 \Leftrightarrow -u_i x_i^* = 0 \tag{11}$$

$$\forall i = 1, \ldots, n, \quad f_i(\mathbf{x}_*) \leq 0 \Leftrightarrow x_i^* \geq 0 \tag{12}$$

$$h_1(\mathbf{x}_*) = 0 \Leftrightarrow \sum_{i=1}^n x_i^* = 1 \tag{13}$$

Let's assume a random integer $k \in [1, n]$. We set $u_k = 0$. Then, from (9) we get that:

$$v = -c_k \tag{14}$$

Moreover, let's assume another random integer $p \in [1, n] \neq k$. Then, from (9) and (14) we get:

$$u_p = c_p - c_k \tag{15}$$

If $c_p < c_k$, (15) violates condition (10). Consequently we must never reach to such a situation and we have to ensure that $c_p \geq c_k$ at all times. The only way to achieve this is to appropriately choose k, in the following way:

$$k \in \left\{\text{argmin}_k(c_k)\right\}$$

Then, for any integer $p \in [1, n]$, condition (10) is satisfied, since $u_p = c_p - c_k \geq 0$. We separate 2 cases:

- If $u_p = 0$, then condition (11) is satisfied too, and x_i^* can get any value as long as it satisfies the remaining conditions (12), (13).
- If $u_p > 0$, then $x_i^* = 0$, in order to satisfy condition (11).

In conclusion, the optimal point $\mathbf{x}_* = [x_1^*, \ldots, x_n^*]^T$ of problem (8) can be computed as follows:

$$\forall i = 1, \ldots, n, \quad x_i^* \begin{cases} = 0, & \text{if } i \notin \left\{\text{argmin}_k(c_k)\right\} \\ \in [0, 1], \text{ s.t. } \sum_{j \in \left\{\text{argmin}_k(c_k)\right\}} x_j^* = 1, & \text{if } i \in \left\{\text{argmin}_k(c_k)\right\} \end{cases}$$

Exercise 4

Questions 4.a, 4.b. On this exercise we have to solve the following optimization problem:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 \tag{16}$$

We generate matrices $\mathbf{A}$, $\mathbf{b}$ twice, by drawing them from the Gaussian and the Uniform distribution. Then, we initially solve the optimization problem via 'CVX' software, and we get the optimal point $\mathbf{x}_*$ and the optimal value f_{opt} (which will be used at the Polyak step below).

It is true that f is a proper closed and convex function. Moreover, $\mathbb{R}^n$ is a nonempty closed and convex set, and f is defined at all of its points. As a result, minimization can take place by using the projected subgradient descent technique, as described in section 8.2.1 of [1]. It's worth observing that:

$$\mathbf{P}_{\mathbb{R}^n}(\mathbf{x}) = \mathbf{x}, \ \forall \mathbf{x} \in \mathbb{R}^n$$

Consequently, the projected subgradient method for the specific problem turns into:

Algorithm 1: Projected subgradient descent algorithm for the solution of (16).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $f'(\mathbf{x}_k) \in \partial f(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - t_k f'(\mathbf{x}_k)$
  $k \leftarrow k + 1$
end

- We choose as stepsize t_k the Polyak step:

$$t_k = \begin{cases} \frac{f(\mathbf{x}_k) - f_{opt}}{\|f'(\mathbf{x}_k)\|^2}, & f'(\mathbf{x}_k) \neq 0 \\ 1, & f'(\mathbf{x}_k) = 0 \end{cases}$$

  where f_{opt} is the solution of the optimization problem (16).

- We use as subgradient $f'(\mathbf{x}_k)$ the following weak result which is proved (if we set $\mathbf{b}_0 = -\mathbf{b}$ on the position of $\mathbf{b}$) by example 3.44 of [1]:

$$f'(\mathbf{x}_k) = \mathbf{A}^T \text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b}) \ \in \partial f(\mathbf{x})$$

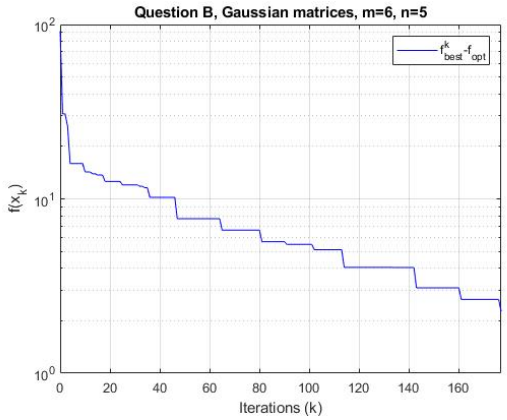
- The iterations terminate when convergence has been achieved. This situation can be detected by the following criterion, for c sufficiently close (and greater than) 1:

$$f(\mathbf{x}_k) \leq c * f_{opt}$$

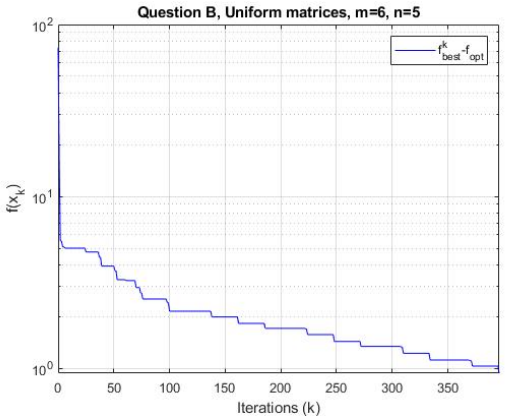
In figure 3a we can see the best (up to the k-th iteration) value of the cost function, f^k_{best}, while it approaches the optimal value f_{opt}. Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Gaussian distribution. Moreover, figure 3b demonstrates again $f^k_{best} - f_{opt}$, but in this case, matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Uniform distribution. Also, c was set to 1.1

in both cases. As expected, in both figures $f^k_{best} - f_{opt}$ converges to 0 as the iterations progress.

On this experiment, we have set $n = 5$ and $m = 6$ ($\mathbf{A} \in \mathbb{R}^{6x5}$, $\mathbf{x} \in \mathbb{R}^5$, $\mathbf{b} \in \mathbb{R}^6$). For $m \gtrapprox n$, we observe that when matrices are drawn from the Gaussian distribution, convergence is faster than in the case of Uniform matrices. Further experiments will be conducted in section 4.*d*.



(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).



(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 3: Projected subgradient descent with Polyak step for $n = 5$ & $m = 6$.

Question 4.c. On this section we will implement algorithm 1 again, but instead of the Polyak step we will use a dynamic stepsize. The main advantage of this technique is that it is not required a prior knowledge of the optimal value f_{opt} in order to estimate the stepsize at each iteration.

More specifically, as dynamic stepsize we will use the following (theorem 8.28 at [1]):

$$t_k = \begin{cases} \frac{1}{\|f'(\mathbf{x}_k)\|\sqrt{k+1}}, & \text{if } \|f'(\mathbf{x}_k)\| \neq 0 \\ \frac{1}{L_f}, & \text{if } \|f'(\mathbf{x}_k)\| = 0 \end{cases}$$

where $L_f \geq \|\mathbf{g}\|_2, \forall \mathbf{g} \in \partial f(\mathbf{x})$.

Let $h(\mathbf{x}) = \|\mathbf{x}\|_1$. From example 3.41 in [1] it can be shown that $\|\partial h(\mathbf{x})\|_2 \leq \sqrt{n}$, for any $\mathbf{x} \in \mathbb{R}^n$. As a result, we get that:

$$f(\mathbf{x}) = h(\mathbf{A}\mathbf{x} - \mathbf{b}) \Rightarrow \partial f(\mathbf{x}) = \mathbf{A}^T \partial h(\mathbf{A}\mathbf{x} - \mathbf{b}) \Rightarrow$$
$$\Rightarrow \|\partial f(\mathbf{x})\|_2 = \|\mathbf{A}^T \partial h(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2 \leq \|\mathbf{A}^T\|_{2,2} \|\partial h(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2 \leq \|\mathbf{A}^T\|_{2,2}\sqrt{m}$$

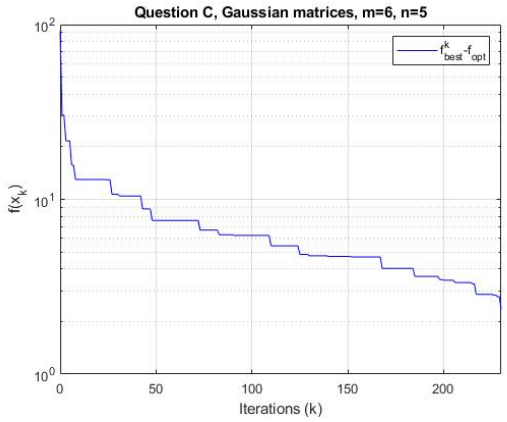
The above result shows that, $\forall \mathbf{g} \in \partial f(\mathbf{x})$, it holds that $\|\mathbf{g}\|_2 \leq \|\mathbf{A}^T\|_{2,2}\sqrt{m}$. This means that we can set:

$$L_f = \|\mathbf{A}^T\|_{2,2}\sqrt{m} = \|\mathbf{A}\|_{2,2}\sqrt{m}$$

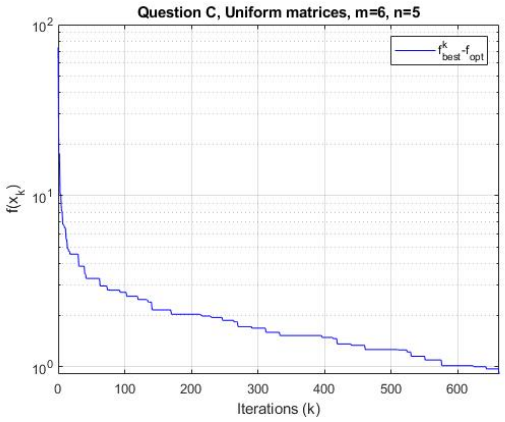
In figure 4a we can see the best (up to the k-th iteration) value of the cost function, f^k_{best}, while it approaches the optimal value f_{opt}. Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Gaussian distribution. Moreover, figure 4b demonstrates again $f^k_{best} - f_{opt}$, but in this case, matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Uniform distribution. Also, in all cases *c*

was set to 1.1. As expected, in both figures $f_{best}^k - f_{opt}$ converges to 0 as the iterations progress.

On this experiment, we have set $n = 5$ and $m = 6$ ($\mathbf{A} \in \mathbb{R}^{6x5}$, $\mathbf{x} \in \mathbb{R}^5$, $\mathbf{b} \in \mathbb{R}^6$). For $m \gtrapprox n$, we observe that convergence is faster at the Gaussian case. Further experiments will be conducted in section 4.*d*.



(a) $f_{best}^k - f_{opt}$ (Gaussian matrices).



(b) $f_{best}^k - f_{opt}$ (Uniform matrices).

Figure 4: Projected subgradient descent with dynamic step for $n = 5$ & $m = 6$.

Questions 4.d, 4.e. On this section we will test the projected subgradient descent algorithm with Polyak and dynamic stepsizes, for various values of m and n. Matrices $\mathbf{A}$ and $\mathbf{b}$ will be generated again twice, as they will be drawn once from the Gaussian and once from the Uniform distribution.

Also, for each figure we will plot an upper bound for the convergence rate of the Polyak stepsize, which is given (according to theorem 8.13 in [1]) by:

$$f_{best}^k - f_{opt} \leq \frac{L_f \left\| \mathbf{x}_0 - \mathbf{x}_* \right\|_2}{\sqrt{k+1}}.$$

Parameter L_f is an upper bound to the norms of the subgradients of function f, and was computed in section 4.*c*. Moreover, the optimal point $\mathbf{x}_*$ has already been computed in section 4.*a* by 'CVX' software.

More specifically:

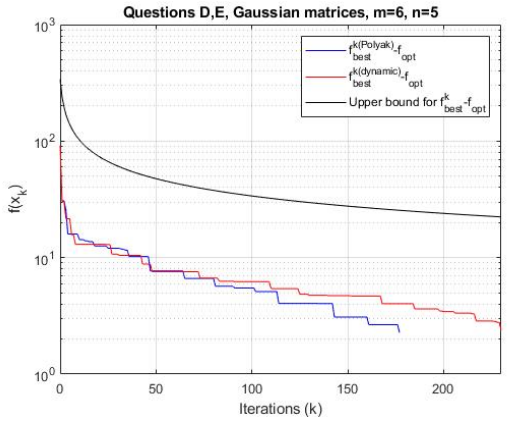
- Figure 5 analyzes the case where $m \gtrapprox n$ ($m = 6$, $n = 5$), by combining the plots of sections 4.*b* and 4.*c*. In what regards the termination criterion, c has been set equal with 1.1. More in detail, plots of fig. 5a were produced with matrices $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Gaussian distribution, while the plots in fig. 5b refer to the case of $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Uniform distribution.

- On the other hand, figure 6 analyzes the case where $m >> n$ ($m = 25$, $n = 5$). In what regards the termination criterion, c has been set equal with 1.01, as the convergence rate for such dimensions is much faster. More specifically, plots of fig. 6a were produced with matrices $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Gaussian distribution, while the plots in fig. 6b refer to the case of $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Uniform distribution.

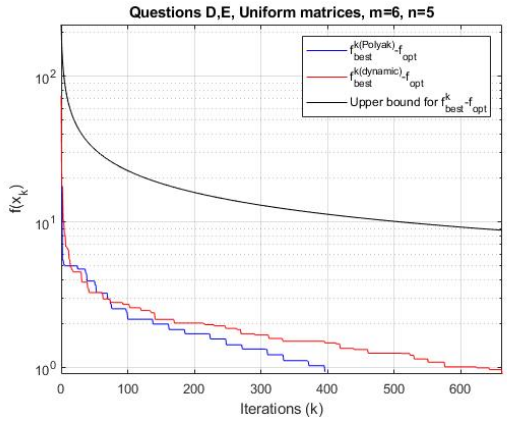
As already known from the theoretical results, the Polyak step is optimal, something which is also proved by all figures. Moreover, when $m >> n$, less iterations are required for convergence, in general.

On the other hand, the performance of the dynamic stepsize varies, depending on the problem. Dynamic stepsizes converge faster when $m >> n$, while they are much slower when $m \gtrapprox n$. Their performance also varies, depending of the type of matrices **A** and **b** (Gaussian or Uniform). However, all algorithms converge remarkably faster than the theoretical upper bound of the Polyak's convergence rate.

Finally, it's worth noticing that, when $m \gtrapprox n$, the performance of the subgradient algorithm with dynamic steps is close to the one with the Polyak step. However, when $m >> n$, the performance of the Polyak step is significantly better than the dynamic steps.

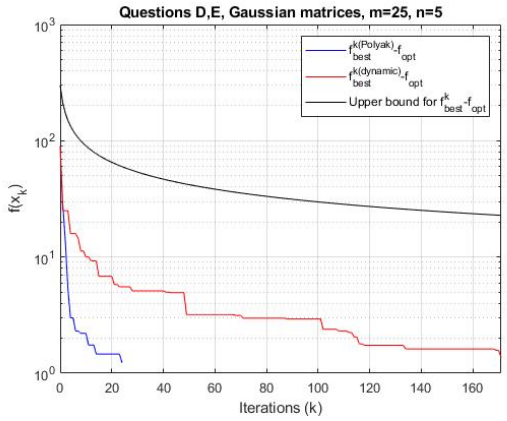


(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).

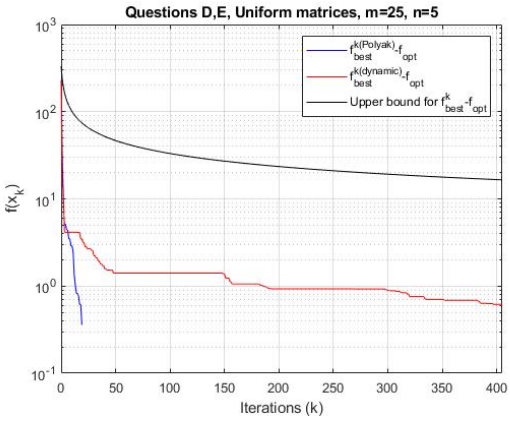


(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 5: Projected subgradient descent for $n = 5$, $m = 6$, $c = 1.1$.



(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).



(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 6: Projected subgradient descent for $n = 5$, $m = 25$, $c = 1.01$.

Question 4.f. The stochastic projected subgradient algorithm is another optimization technique, variant of the projected subgradient descent algorithm. Again, we will utilize it to solve the optimization problem (16).

Let's denote with $\mathbf{a}_i^T$ the i-th row of $\mathbf{A}$. Then, if we set $f_i(\mathbf{x}) = |\mathbf{a}_i^T\mathbf{x} - b_i|$, it occurs that:

$$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 = \sum_{i=1}^{n} |\mathbf{a}_i^T\mathbf{x} - b_i| = \sum_{i=1}^{n} f_i(\mathbf{x})$$

The subdifferential set of functions f_i is:

$$\partial f_i(\mathbf{x}) = \begin{cases} \mathbf{a}_i, & \text{if } \mathbf{a}_i^T\mathbf{x} - b_i > 0 \\ -\mathbf{a}_i, & \text{if } \mathbf{a}_i^T\mathbf{x} - b_i < 0 \\ \xi\mathbf{a}_i : \xi \in [-1,1], & \text{if } \mathbf{a}_i^T\mathbf{x} - b_i = 0 \end{cases} \tag{17}$$

The iterations of the algorithm will be divided in epochs, with each epoch consisting of m iterations. Moreover, $\mathbf{P}_{\mathbb{R}^n}(\mathbf{x}) = \mathbf{x},\ \forall \mathbf{x} \in \mathbb{R}^n$, as explained in section 4.*a*, 4.*b*. Also, we define a constant Θ to be an upper bound on the half-squared diameter of the subset of $\mathbb{R}^n$ from which $\mathbf{x}$ will get its values. More specifically, $\mathbf{x}_k$ will be closer than $\mathbf{x}_0$ to the optimal point $\mathbf{x}_*$, for any $k \geq 1$, as $f(\mathbf{x})$ is convex. That means:

$$\Theta \geq \frac{1}{2}\left(2\|\mathbf{x}_0 - \mathbf{x}_*\|_2^2\right) = \|\mathbf{x}_0 - \mathbf{x}_*\|_2^2$$

Furthermore, we denote again with L_f a constant such that $L_f \geq \|\mathbf{g}\|_2, \forall \mathbf{g} \in \partial f(\mathbf{x})$. From eq. (17) it easily occurs that $\|\partial f_i(\mathbf{x})\|_2 \leq \|\mathbf{a}_i\|_2, \forall i = 1, \ldots, n$. So, we can set:

$$L_{f_i} = \|\mathbf{a}_i\|_2$$

Moreover, we denote with L_f the following quantity:

$$L_f = \sqrt{m}\sqrt{\sum_{i=1}^{m} L_{f_i}^2} = \sqrt{m}\sqrt{\sum_{i=1}^{m} \|\mathbf{a}_i\|_2^2} = \sqrt{m}\|\mathbf{A}\|_{2,2}$$

Notice that L_f here is equal with the respective quantity computed at section 4.*c* (as expected, since the same cost function is utilized).

Consequently, the algorithm that is presented in section 8.3.2 of [1] turns into:

Algorithm 2: Stochastic projected subgradient algorithm that solves (16).

```
Choose a random x_0 ∈ R^n
Θ ← ||x_0 − x_*||_2^2
for e=1:epochs do
    for j=1:m do
        k ← (e − 1)m + j − 1
        Draw i_k ∈ {1, ..., m} uniformly, and choose any f'_{i_k}(x_k) ∈ ∂f_{i_k}(x_k)
        x_{k+1} ← x_k − (√(2Θ)m / (L_f √(k+1))) f'_{i_k}(x_k)
    end
end
```

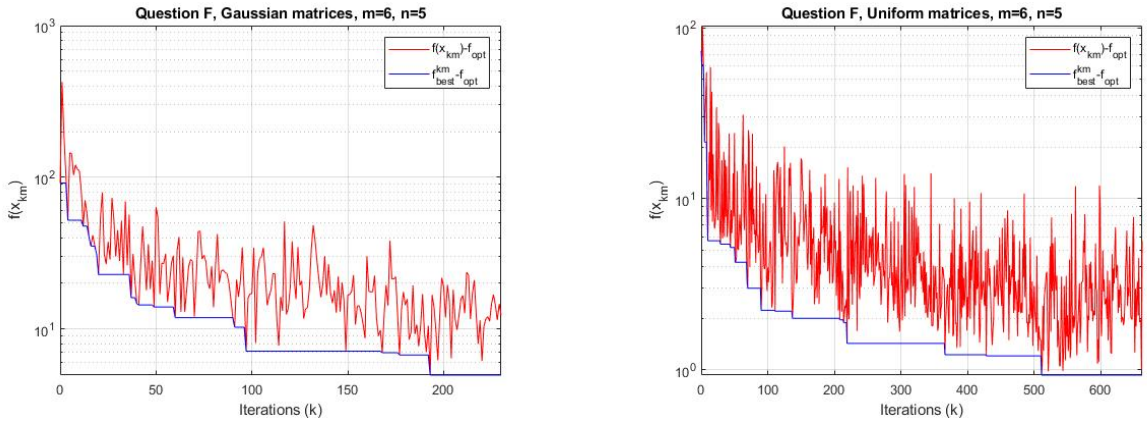
In figures 7 we plot the value $f(\mathbf{x}_{km}) - f_{opt}$ of the cost function every m iterations. We execute $(K_{dyn}m)$ iterations in total, where K_{dyn} the number of iterations until the respective algorithm of section 4.*c* terminates.

In figure 7a we can see the best (up to the k-th iteration) value of the cost function, f_{best}^{km} (blue), while it approaches the optimal value f_{opt}. Furthermore, we can see the

difference of the actual value of the cost function $f(\mathbf{x}_{km})$ (red) from the optimal value. Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Gaussian distribution.

Moreover, figure 7b demonstrates again $f_{best}^{km} - f_{opt}$ and $f(\mathbf{x}_{km}) - f_{opt}$, but in this case, matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Uniform distribution. Also, in all cases c was set to 1.1. As expected, in both figures f_{best}^{km} converges to f_{opt} as the iterations progress.

On this experiment, we have set $n = 5$ and $m = 6$ ($\mathbf{A} \in \mathbb{R}^{6x5}$, $\mathbf{x} \in \mathbb{R}^5$, $\mathbf{b} \in \mathbb{R}^6$). Again, for $m \gtrapprox n$ we observe that convergence is faster at the Gaussian case. Further experiments will be conducted in section 4.*h*.



(a) $f_{best}^{km} - f_{opt}$, $f(\mathbf{x}_{km}) - f_{opt}$ (Gaus. mat.). (b) $f_{best}^{km} - f_{opt}$, $f(\mathbf{x}_{km}) - f_{opt}$ (Unif. mat.).

Figure 7: Stochastic subgradient descent for $n = 5$, $m = 6$, $c = 1.1$.

Question 4.g. The incremental projected subgradient algorithm is similar to the stochastic one, with the difference that the random part is replaced by a deterministic. If we define all parameters the same way as in section 4.*f*, the algorithm (which is presented in section 8.4 of [1]) turns into:

Algorithm 3: Incremental projected subgradient algorithm that solves (16).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$
$\Theta \leftarrow \|\mathbf{x}_0 - \mathbf{x}_*\|_2^2$
for *k=0:epochs-1* **do**
  $\mathbf{x}_{k,0} \leftarrow \mathbf{x}_k$
  for *j=1:m* **do**
    $\mathbf{x}_{k,j} \leftarrow \mathbf{x}_{k,j-1} - \frac{\sqrt{2\Theta}m}{L_f\sqrt{k+1}} f_j'(\mathbf{x}_{k,j-1})$, where $f_j'(\mathbf{x}_{k,j-1}) \in \partial f_j(\mathbf{x}_{k,j-1})$
  end
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k,m}$
end

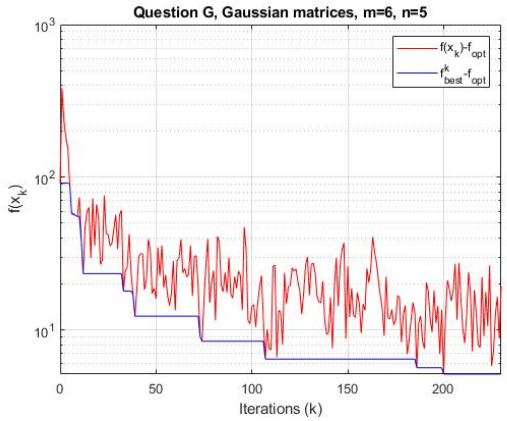
In figures 8 we plot the value $f(\mathbf{x}_k) - f_{opt}$ of the cost function every m iterations. We execute $(K_{dyn}m)$ iterations in total, where K_{dyn} the number of iterations until the respective algorithm of section 4.*c* terminates.

In figure 8a we can see the best (up to the k-th outer iteration) value of the cost function, f_{best}^k (blue), while it approaches the optimal value f_{opt}. Furthermore, we can see the difference of the actual from the optimal value of the cost function $f(\mathbf{x}_k) - f_{opt}$ (red). Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Gaussian distribution.

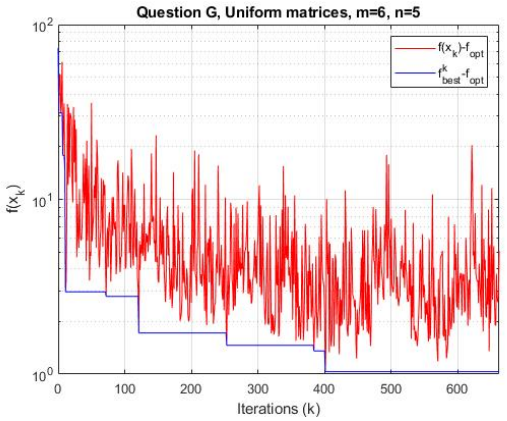
Moreover, figure 8b demonstrates again $f_{best}^k - f_{opt}$ and $f(\mathbf{x}_k) - f_{opt}$, but in this case matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Uniform distribution. Also, in all cases c was set to

1.1. As expected, in both figures f_{best}^k converges to f_{opt} as the iterations progress.

On this experiment, we have set $n = 5$ and $m = 6$ ($\mathbf{A} \in \mathbb{R}^{6x5}$, $\mathbf{x} \in \mathbb{R}^5$, $\mathbf{b} \in \mathbb{R}^6$). Again, for $m \gtrapprox n$ we observe that convergence is faster at the Gaussian case. Further experiments will be conducted in section 4.*h*.



(a) $f_{best}^k - f_{opt}$, $f(\mathbf{x}_k) - f_{opt}$ (Gaus. mat.).



(b) $f_{best}^k - f_{opt}$, $f(\mathbf{x}_k) - f_{opt}$ (Unif. mat.).

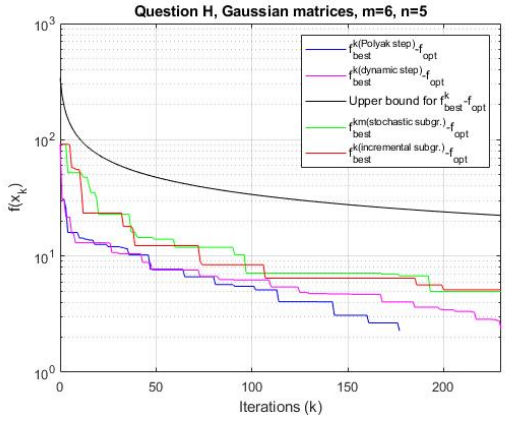
Figure 8: Incremental subgradient descent for $n = 5$, $m = 6$, $c = 1.1$.

Question 4.h. On this section we will compare the performance of some variants of the projected subgradient descent algorithm, for various values of m and n. Matrices $\mathbf{A}$ and $\mathbf{b}$ will be generated again twice, as they will be drawn once from the Gaussian and once from the Uniform distribution. More specifically:

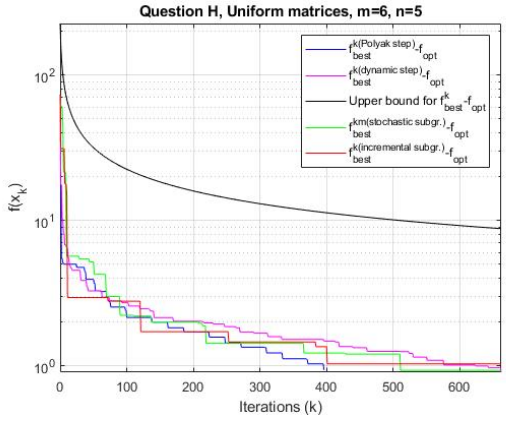
- Figure 9 analyzes the case where $m \gtrapprox n$ ($m = 6, n = 5$), by combining the plots of sections 4.*d*, *e*, 4.*f* and 4.*g*. There have been executed as many (outer) iterations as in the respective diagrams from section 4.*c*. More in detail, plots of fig. 9a were produced with matrices $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Gaussian distribution, while the plots in fig. 9b refer to the case of $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Uniform distribution.

- On the other hand, figure 10 analyzes the case where $m >> n$ ($m = 25, n = 5$). Again, the number of iterations is the same with the respective diagrams from section 4.*c*. More specifically, plots of fig. 10a were produced with matrices $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Gaussian distribution, while the plots in fig. 10b refer to the case of $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Uniform distribution. On this case, parameter c of the termination criterion was set to 1.01, since the iterations converge much faster.

The figures verify once more the optimality of Polyak step, since the respective algorithm converges faster than the rest. We make the same conclusions as in section 4.*d* for both the stochastic and the incremental subgradient algorithm. Notice that their performance is about the same compared to each other. Moreover, we observe again that all algorithms achieve better performance than the theoretical upper bound of the Polyak convergence rate.

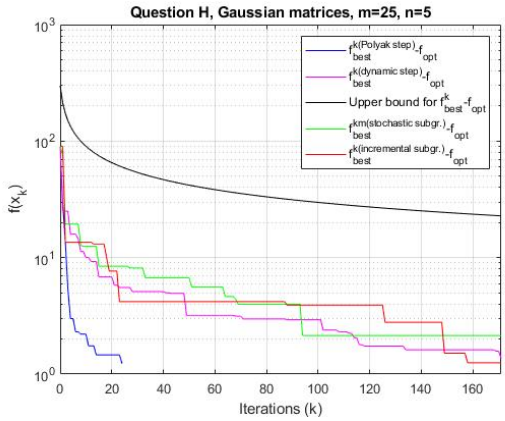


(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).

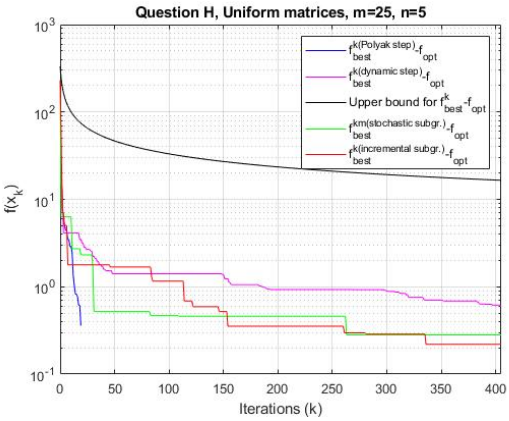


(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 9: Variants of subgradient algorithm for $n = 5$, $m = 6$, $c = 1.1$.



(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).



(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 10: Variants of subgradient algorithm for $n = 5$, $m = 25$, $c = 1.01$.

Question 4.i. The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex04.m`.

Exercise 5

Questions 5.a, 5.b. On this exercise we have to solve the following optimization problem:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_\infty \tag{18}$$

We generate matrices $\mathbf{A}$, $\mathbf{b}$ twice, by drawing them from the Gaussian and the Uniform distribution. Then, we initially solve the optimization problem via 'CVX' software, and we get the optimal point $\mathbf{x}_*$ and the optimal value f_{opt} (which will be used at the Polyak step below).

It is true that f is a proper closed and convex function. Moreover, $\mathbb{R}^n$ is a nonempty closed and convex set, and f is defined at all of its points. As a result, minimization can take place by using the projected subgradient descent technique, as described in section 8.2.1 of [1]. It's worth observing that:

$$\mathbf{P}_{\mathbb{R}^n}(\mathbf{x}) = \mathbf{x}, \ \forall \mathbf{x} \in \mathbb{R}^n$$

Consequently, the projected subgradient method for the specific problem turns into:

Algorithm 4: Projected subgradient descent algorithm for the solution of (18).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $f'(\mathbf{x}_k) \in \partial f(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - t_k f'(\mathbf{x}_k)$
  $k \leftarrow k + 1$
end

- We choose as stepsize t_k the Polyak step:

$$t_k = \begin{cases} \frac{f(\mathbf{x}_k) - f_{opt}}{\|f'(\mathbf{x}_k)\|^2}, & f'(\mathbf{x}_k) \neq 0 \\ 1, & f'(\mathbf{x}_k) = 0 \end{cases}$$

  where f_{opt} is the solution of the optimization problem (18).

- According to example 3.54 in [1] (if we set $\mathbf{b}_0 = -\mathbf{b}$ on the position of $\mathbf{b}$), the subdifferential set of the cost function f is:

$$\partial f(\mathbf{x}) = \begin{cases} \mathbf{A}^T B_{\|.\|_1}[0,1] = \left\{ \mathbf{A}^T \mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_1 \leq 1 \right\}, & \text{if } \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \\ \left\{ \sum_{i \in I_\mathbf{x}} \lambda_i \text{sgn}\left(\mathbf{a}_i^T \mathbf{x} - b_i\right) \mathbf{a}_i : \sum_{i \in I_\mathbf{x}} \lambda_i = 1, \lambda_j \geq 0, j \in \mathbf{I}_\mathbf{x} \right\}, & \text{if } \mathbf{A}\mathbf{x} - \mathbf{b} \neq \mathbf{0} \end{cases}$$

  where with $\mathbf{a}_i^T$ is denoted the i-th line of matrix $\mathbf{A}$ and with $B_{\|.\|_1}[0,1]$ is denoted the ℓ_1-norm unit ball set. Moreover, in what regards the set $I_\mathbf{x}$, we have:

$$I_\mathbf{x} = \left\{ i \in \{1, \ldots, m\} : \left| \mathbf{a}_i^T \mathbf{x} - b_i \right| = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_\infty \right\}$$

  We use as subgradient $f'(\mathbf{x}_k)$ the following result which is proved if we set $\lambda_i = 1$ and $\lambda_j = 0$, $\forall j \neq i$:

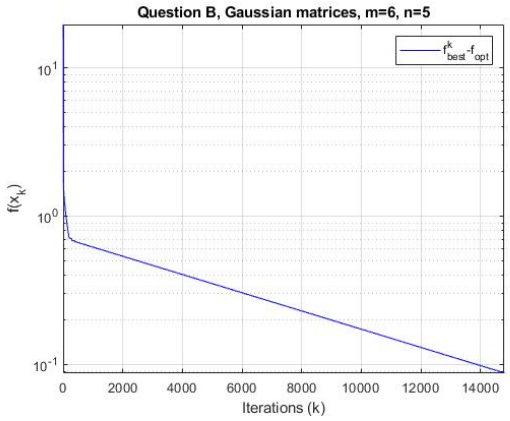
$$f'(\mathbf{x}_k) = \begin{cases} \mathbf{0} \in \mathbb{R}^n, & \text{if } \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \\ \text{sgn}\left(\mathbf{a}_i^T \mathbf{x} - b_i\right) \mathbf{a}_i : i \in I_\mathbf{x}, & \text{if } \mathbf{A}\mathbf{x} - \mathbf{b} \neq \mathbf{0} \end{cases}$$

- The iterations terminate when convergence has been achieved. This situation can be detected by the following criterion, for c sufficiently close (and greater than) 1:

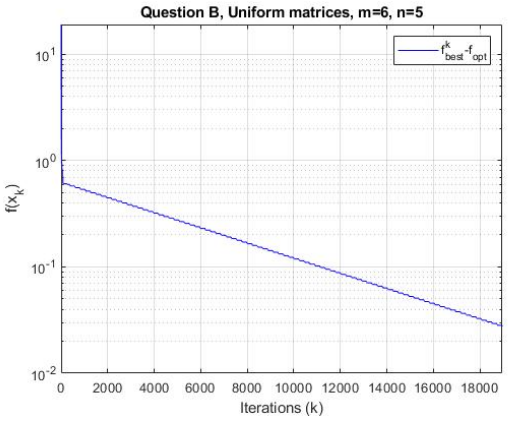
$$f(\mathbf{x}_k) \leq c * f_{opt}$$

In figure 11a we can see the best (up to the k-th iteration) value of the cost function, f_{best}^k, while it approaches the optimal value f_{opt}. Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Gaussian distribution. Moreover, figure 11b demonstrates again $f_{best}^k - f_{opt}$, but in this case, matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Uniform distribution. Also, c was set to 1.01 in both cases. As expected, in both figures f_{best}^k converges to f_{opt} as the iterations progress.

On this experiment, we have set $n = 5$ and $m = 6$ ($\mathbf{A} \in \mathbb{R}^{6x5}$, $\mathbf{x} \in \mathbb{R}^5$, $\mathbf{b} \in \mathbb{R}^6$). For $m \gtrapprox n$, we observe that the convergence rate is about the same for the different types of matrices, with the Gaussian matrices converging slightly faster than the Uniform ones. Further experiments will be conducted in section 5.*d*.



(a) $f_{best}^k - f_{opt}$ (Gaussian matrices).



(b) $f_{best}^k - f_{opt}$ (Uniform matrices).

Figure 11: Projected subgradient descent with Polyak step for $n = 5$ & $m = 6$.

Question 5.c. On this section we will implement algorithm 4 again, but instead of the Polyak step we will use a dynamic stepsize. The main advantage of this technique is that it is not required a prior knowledge of the optimal value f_{opt} in order to estimate the stepsize at each iteration.

More specifically, as dynamic stepsize we will use the following (theorem 8.28 at [1]):

$$t_k = \begin{cases} \frac{1}{\|f'(\mathbf{x}_k)\|\sqrt{k+1}}, & \text{if } \|f'(\mathbf{x}_k)\| \neq 0 \\ \frac{1}{L_f}, & \text{if } \|f'(\mathbf{x}_k)\| = 0 \end{cases}$$

where $L_f \geq \|\mathbf{g}\|_2, \forall \mathbf{g} \in \partial f(\mathbf{x})$.

Let $h(\mathbf{x}) = \|\mathbf{x}\|_\infty$. Then, from example 3.52 in [1] and if $\mathbf{x} \neq \mathbf{0}$, $\forall \mathbf{g}_h \in \partial h(\mathbf{x})$ it holds that:

$$\|\mathbf{g}_h\|_2 = \left\| \sum_{i \in I_\mathbf{x}} \lambda_i \text{sgn}(x_i)\, \mathbf{e}_i \right\|_2 \leq \sum_{i \in I_\mathbf{x}} \lambda_i |\text{sgn}(x_i)|\, \|\mathbf{e}_i\|_2 = \sum_{i \in I_\mathbf{x}} \lambda_i \sqrt{n} = \sqrt{n}$$

Moreover, if $\mathbf{x} = 0$ then $\mathbf{g}_h \in B_{\|.\|_1}[0,1]$ and

$$\|\mathbf{g}_h\|_2 \leq \|\mathbf{g}_h\|_1 \leq 1 \leq \sqrt{n}, \quad \forall\, n \geq 1$$

As a result, we get that:

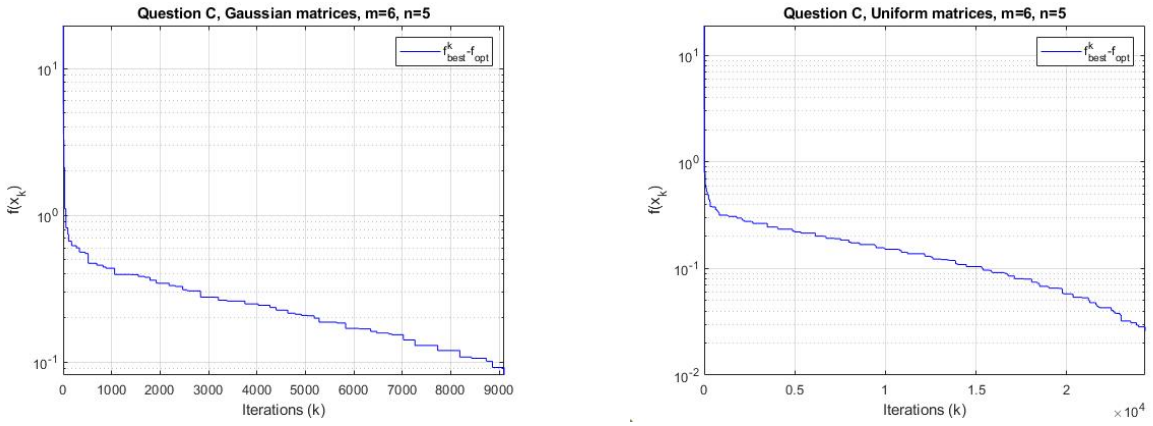
$$f(\mathbf{x}) = h(\mathbf{A}\mathbf{x} - \mathbf{b}) \Rightarrow \partial f(\mathbf{x}) = \mathbf{A}^T \partial h(\mathbf{A}\mathbf{x} - \mathbf{b}) \Rightarrow$$
$$\Rightarrow \|\partial f(\mathbf{x})\|_2 = \|\mathbf{A}^T \partial h(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2 \leq \|\mathbf{A}^T\|_{2,2} \|\partial h(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2 \leq \|\mathbf{A}^T\|_{2,2}\sqrt{m}$$

The above result shows that, $\forall \mathbf{g} \in \partial f(\mathbf{x})$, it holds that $\|\mathbf{g}\|_2 \leq \|\mathbf{A}^T\|_{2,2}\sqrt{m}$. This means that we can set

$$L_f = \|\mathbf{A}^T\|_{2,2}\sqrt{m} = \|\mathbf{A}\|_{2,2}\sqrt{m}$$

In figure 12a we can see the best (up to the k-th iteration) value of the cost function, f^k_{best}, while it approaches the optimal value f_{opt}. Matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Gaussian distribution. Moreover, figure 12b demonstrates again $f^k_{best} - f_{opt}$, but in this case, matrices $\mathbf{A}$, $\mathbf{b}$ were drawn from the Uniform distribution. Also, in all cases c was set to 1.01. As expected, in both figures f^k_{best} converges to f_{opt} as the iterations progress.

On this experiment, we have set $n = 5$ and $m = 6$ ($\mathbf{A} \in \mathbb{R}^{6x5}$, $\mathbf{x} \in \mathbb{R}^5$, $\mathbf{b} \in \mathbb{R}^6$). For $m \gtrapprox n$, we observe that convergence is significantly faster at the Gaussian case. Further experiments will be conducted in section 5.*d*.



(a) $f^k_{best} - f_{opt}$ (Gaussian matrices). (b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 12: Projected subgradient descent with dynamic step for $n = 5$ & $m = 6$.

Questions 5.d, 5.e. On this section we will test the projected subgradient descent algorithm with Polyak and dynamic stepsizes, for various values of m and n. Matrices $\mathbf{A}$ and $\mathbf{b}$ will be generated again twice, as they will be drawn once from the Gaussian and once from the Uniform distribution.

Also, for each figure we will plot an upper bound for the convergence rate of the Polyak stepsize, which is given (according to theorem 8.13 in [1]) by:

$$f^k_{best} - f_{opt} \leq \frac{L_f \|\mathbf{x}_0 - \mathbf{x}_*\|_2}{\sqrt{k+1}}.$$

Parameter L_f is an upper bound to the norms of the subgradients of function f, and was computed in section 5.*c*. Moreover, the optimal point $\mathbf{x}_*$ has already been computed in section 5.*a* by 'CVX' software.

More specifically:

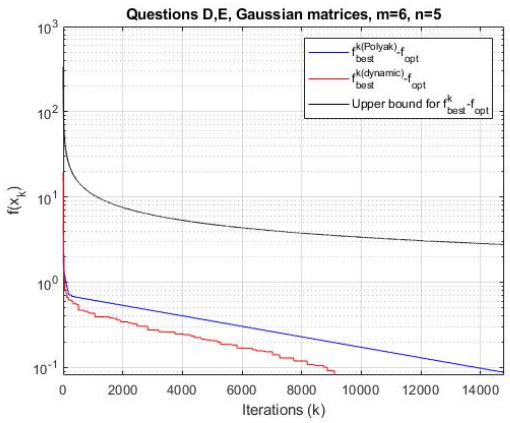
- Figure 13 analyzes the case where $m \gtrapprox n$ ($m = 6$, $n = 5$), by combining the plots of sections 5.*b* and 5.*c*. In what regards the termination criterion, c has been set equal with 1.01. More in detail, plots of fig. 13a were produced with matrices $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Gaussian distribution, while the plots in fig. 13b refer to the case of $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Uniform distribution.

- On the other hand, figure 14 analyzes the case where $m >> n$ ($m = 25$, $n = 5$). In what regards the termination criterion, c has been set equal with 1.01 again. More specifically, plots of fig. 14a were produced with matrices $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Gaussian distribution, while the plots in fig. 14b refer to the case of $\mathbf{A}$, $\mathbf{b}$ having been drawn from the Uniform distribution.

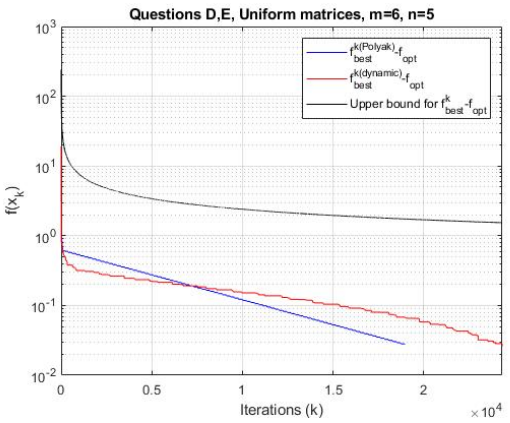
As already known from the theoretical results, the Polyak step is optimal, something which is proved by most figures. Figure 13a consists the only exception, where the iterations with the Polyak step converge slower than the iterations with the dynamic step. The reason for that is mainly luck; for the specific problem, initial point and parameters, the dynamic step is lucky enough to converge to the optimal point faster than the Polyak step.

Another observation we make by comparing the figures is that, when $m >> n$, less iterations are required for convergence. The same conclusion was also made in exercise 4. In what regards the different types of matrices (Gaussian or Uniform), the performance of the iterations with the dynamic step seems to be affected more, compared to the performance of the iterations with the Polyak step. However, all algorithms converge remarkably faster than the theoretical upper bound of the Polyak's convergence rate.

Finally, it's worth noticing that in this exercise, the performance of the subgradient algorithm with dynamic steps is relatively close to the one with the Polyak step, in general, irrespectively of the dimensions m, n.

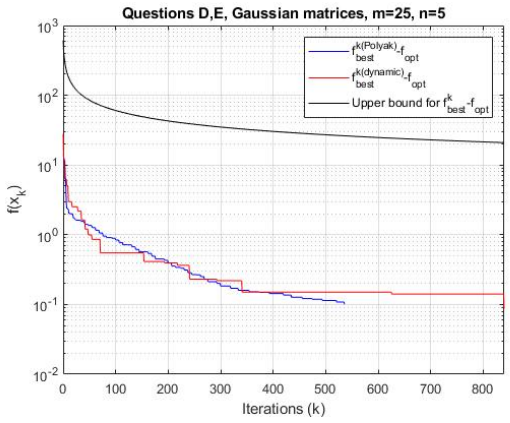


(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).

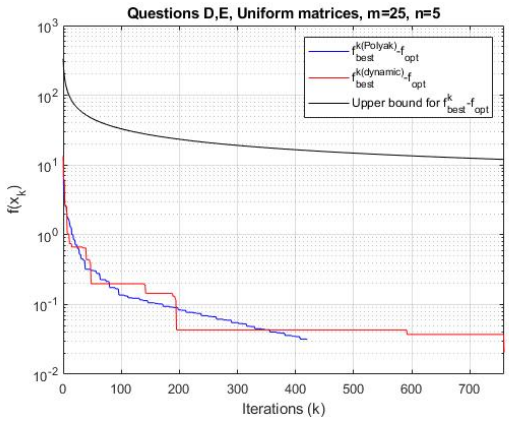


(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 13: Projected subgradient descent for $n = 5$, $m = 6$, $c = 1.01$.



(a) $f^k_{best} - f_{opt}$ (Gaussian matrices).



(b) $f^k_{best} - f_{opt}$ (Uniform matrices).

Figure 14: Projected subgradient descent for $n = 5$, $m = 25$, $c = 1.01$.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex05.m`.

Exercise 6

Question 6.a. As a first step, we set the dimensions m, n, s of the problem, under the requirements defined by the exercise. We decide to set $n = 128$ and $s = 4 << n$. Then $m \gtrapprox 2s \log n = 38.82$, so we set $m = 40 << n$.

In fig. 15a, 15b, 16 we can see the optimal points $\mathbf{x}_{1(opt)}$, $\mathbf{x}_{2(opt)}$ of functions f_1, f_2 respectively, as they were computed by 'CVX' software for various values of parameter λ.With blue highlighting are marked the elements of the (sparse) vector $\mathbf{x}_s$ which are different than 0. Also, in table 1 we can see the (optimal) values of functions f_1, f_2, at those points.

	xs	x1, λ=0.01	x1, λ=0.1	x1, λ=1	x1, λ=10	x2, λ=0.01	x2, λ=0.1	x2, λ=1	x2, λ=10
1	0	8.6588e-08	2.1074e-08	6.4232e-09	2.8859e-08	-0.1900	-0.1751	-0.0659	0.1241
2	0	2.3142e-08	5.5675e-09	1.6855e-09	7.5657e-09	-0.0708	-0.0544	0.0384	0.1445
3	0	1.8319e-08	4.4210e-09	1.3345e-09	5.9940e-09	0.4979	0.4797	0.3727	0.2236
4	0	3.3711e-08	8.1782e-09	2.4720e-09	1.1055e-08	0.9890	0.9619	0.7761	0.3671
5	0	1.5679e-08	3.8058e-09	1.1502e-09	5.1494e-09	-0.1131	-0.1125	-0.1049	-0.0056
6	0	1.0069e-08	2.4339e-09	7.3480e-10	3.2985e-09	-0.6203	-0.6002	-0.4524	-0.0730
7	0	3.0902e-07	6.5332e-08	1.9760e-08	9.4508e-08	0.5369	0.5342	0.5110	0.3706
8	0	3.2528e-08	7.8799e-09	2.3899e-09	1.0773e-08	-0.0988	-0.1033	-0.1097	0.0111
9	0	2.2082e-08	5.3221e-09	1.6062e-09	7.2106e-09	0.4964	0.4833	0.4001	0.2434
10	0	1.6372e-08	3.9883e-09	1.2062e-09	5.3934e-09	0.0341	0.0155	-0.0719	-0.0221
11	0	6.3378e-08	1.5526e-08	4.6921e-09	2.0943e-08	1.0508	1.0172	0.8106	0.4191
12	0	2.7212e-08	6.5189e-09	1.9679e-09	8.8664e-09	0.8694	0.8722	0.8358	0.4763
13	0	3.6609e-08	8.8704e-09	2.6856e-09	1.2015e-08	0.0219	0.0373	0.1216	0.2046
14	0	1.9181e-08	4.6851e-09	1.4149e-09	6.3209e-09	0.4478	0.4242	0.2817	0.1137
15	0	2.3762e-08	5.7783e-09	1.7494e-09	7.8330e-09	-0.6006	-0.5847	-0.4586	-0.0721
16	0	1.9596e-04	0.0020	0.0197	0.1975	-0.8810	-0.8404	-0.5733	-0.0479
17	0	1.2565e-08	3.0197e-09	9.1212e-10	4.0960e-09	0.1721	0.1692	0.1486	0.1216
18	0	2.9722e-08	7.2859e-09	2.2053e-09	9.8375e-09	-0.6294	-0.5980	-0.3996	-0.0289
19	0	1.0719e-08	2.5622e-09	7.7283e-10	3.4784e-09	0.2761	0.2646	0.1790	0.0652
20	0	1.8246e-08	4.4138e-09	1.3363e-09	6.0024e-09	-0.4071	-0.4061	-0.3862	-0.1430
21	0	1.0652e-08	2.5679e-09	7.7470e-10	3.4800e-09	0.0278	0.0189	-0.0344	-0.0032
22	0	1.5751e-08	3.7638e-09	1.1353e-09	5.1104e-09	0.2622	0.2482	0.1837	0.1585
23	0	1.6501e-08	3.9196e-09	1.1806e-09	5.3323e-09	0.2696	0.2880	0.3860	0.3643
24	0	4.0809e-08	9.8977e-09	2.9909e-09	1.3398e-08	0.5393	0.5537	0.5849	0.3631
25	0	2.5311e-05	2.6050e-04	0.0026	0.0261	0.7527	0.7383	0.6394	0.3900
26	0	2.5369e-08	6.1262e-09	1.8513e-09	8.3168e-09	-0.1235	-0.1215	-0.0880	0.0696
27	0	4.4545e-08	1.0792e-08	3.2800e-09	1.4716e-08	-0.4550	-0.4251	-0.2333	0.0751
28	0	1.0149e-07	2.4373e-08	7.3655e-09	3.3219e-08	0.7392	0.7410	0.7228	0.4665
29	7.5727	7.5721	7.5660	7.5052	6.8979	2.7300	2.6742	2.2542	1.0376
30	0	3.8068e-08	9.1616e-09	2.7760e-09	1.2481e-08	0.2290	0.2437	0.3191	0.3154
31	0	1.5365e-08	3.7364e-09	1.1292e-09	5.0521e-09	0.2166	0.2033	0.1112	0.0447
32	0	2.3954e-06	9.3045e-07	1.2639e-07	9.6378e-07	0.3559	0.3672	0.4206	0.3561
33	0	6.5621e-04	0.0066	0.0656	0.6564	1.3048	1.2900	1.1476	0.6315
34	0	2.9426e-08	7.1627e-09	2.1672e-09	9.7041e-09	-0.1354	-0.1303	-0.0847	0.0744
35	0	1.9135e-08	4.6481e-09	1.4094e-09	6.3194e-09	-1.0371	-1.0015	-0.7587	-0.1765
36	0	5.6892e-08	1.4047e-08	4.2524e-09	1.8813e-08	0.4144	0.4068	0.3658	0.2580
37	0	2.7194e-08	6.5201e-09	1.9731e-09	8.8972e-09	0.1309	0.1339	0.1462	0.1504
38	0	1.0734e-08	2.5608e-09	7.7136e-10	3.4776e-09	0.2241	0.2309	0.2631	0.2147
39	0	6.4584e-08	1.5689e-08	4.7554e-09	2.1355e-08	0.1108	0.1173	0.1555	0.1972
40	0	1.5278e-08	3.6458e-09	1.0994e-09	4.9488e-09	0.9472	0.9182	0.7277	0.3451
41	0	2.2881e-08	5.5459e-09	1.6734e-09	7.5017e-09	-0.0432	-0.0242	0.0806	0.1603
42	0	3.1374e-08	7.6354e-09	2.3129e-09	1.0393e-08	-0.7232	-0.7082	-0.5709	-0.1256
43	0	2.6351e-08	6.3789e-09	1.9313e-09	8.6471e-09	-0.4043	-0.3722	-0.1829	0.0985
44	0	1.9271e-08	4.6713e-09	1.4111e-09	6.3272e-09	-0.5274	-0.5117	-0.3879	-0.0258
45	0	6.7771e-08	1.6623e-08	5.0474e-09	2.2495e-08	0.5339	0.5051	0.3305	0.1263
46	0	2.2306e-08	5.3419e-09	1.6133e-09	7.2568e-09	0.3597	0.3513	0.2969	0.2091
47	0	2.0843e-08	5.0215e-09	1.5209e-09	6.8549e-09	0.1109	0.0991	0.0385	0.0796
48	0	4.9545e-08	1.1915e-08	3.6209e-09	1.6462e-08	-0.7689	-0.7495	-0.6000	-0.1599
49	0	1.9132e-08	4.6011e-09	1.3888e-09	6.2386e-09	0.4792	0.4533	0.3177	0.1886
50	0	5.8435e-08	1.4263e-08	4.3317e-09	1.9310e-08	-0.7028	-0.6553	-0.3661	0.0625
51	0	1.4661e-08	3.5533e-09	1.0734e-09	4.8096e-09	-0.1715	-0.1620	-0.1109	0.0409
52	0	5.0825e-08	1.2255e-08	3.7317e-09	1.6859e-08	-0.5967	-0.5795	-0.4405	-0.0389
53	0	3.6970e-04	0.0037	0.0371	0.3708	0.3624	0.3865	0.5131	0.4566
54	0	1.6190e-08	3.9089e-09	1.1833e-09	5.3076e-09	-0.2635	-0.2698	-0.2668	-0.0300
55	0	7.0505e-08	1.6966e-08	5.1541e-09	2.3501e-08	0.9916	0.9649	0.7752	0.3646
56	0	1.5627e-07	3.5715e-08	1.0888e-08	5.1639e-08	0.0905	0.0941	0.1074	0.1398
57	0	2.6460e-08	6.3560e-09	1.9268e-09	8.7000e-09	-0.4471	-0.4201	-0.2527	0.0430
58	0	2.5595e-08	6.1050e-09	1.8486e-09	8.3277e-09	-0.0391	-0.0097	0.1601	0.2729

(a) Elements $1 - 58$ of the optimal points.

	xs	x1, λ=0.01	x1, λ=0.1	x1, λ=1	x1, λ=10	x2, λ=0.01	x2, λ=0.1	x2, λ=1	x2, λ=10
59	0	5.3972e-08	1.3119e-08	3.9938e-09	1.7940e-08	-0.6356	-0.6177	-0.4703	-0.0577
60	3.5739	3.5731	3.5661	3.4957	2.7919	1.6213	1.5848	1.3255	0.6237
61	0	1.5266e-08	3.6641e-09	1.1046e-09	4.9717e-09	-0.1431	-0.1371	-0.0737	0.1117
62	7.0100	7.0092	7.0026	6.9365	6.2753	2.6661	2.5915	2.0693	0.8429
63	0	2.2666e-08	5.5349e-09	1.6722e-09	7.4599e-09	-0.1273	-0.1183	-0.0544	0.1049
64	0	2.5553e-08	6.1734e-09	1.8708e-09	8.4037e-09	0.2101	0.1927	0.0899	0.0403
65	0	1.5334e-08	3.6885e-09	1.1103e-09	4.9959e-09	0.3421	0.3366	0.2894	0.1825
66	0	2.0396e-07	4.5436e-08	1.3833e-08	6.5076e-08	0.4643	0.4634	0.4338	0.2711
67	0	1.6657e-08	4.0163e-09	1.2131e-09	5.4425e-09	0.3006	0.2847	0.2077	0.1506
68	5.4669	5.4666	5.4639	5.4367	5.1642	1.9371	1.8746	1.4524	0.5898
69	0	2.1063e-08	5.1260e-09	1.5506e-09	6.9341e-09	-0.6207	-0.6057	-0.4777	-0.0733
70	0	1.8016e-08	4.3054e-09	1.2997e-09	5.8507e-09	0.2266	0.2364	0.2940	0.3003
71	0	5.2629e-08	1.2535e-08	3.8237e-09	1.7528e-08	-0.3199	-0.3241	-0.3253	-0.0793
72	0	2.4002e-08	5.7778e-09	1.7477e-09	7.8435e-09	0.6422	0.6326	0.5620	0.3253
73	0	3.6313e-08	8.8873e-09	2.6846e-09	1.1972e-08	-0.0136	9.4884e-04	0.0873	0.1775
74	0	9.0265e-08	2.2098e-08	6.6943e-09	2.9867e-08	0.3145	0.3210	0.3500	0.3111
75	0	4.0401e-08	9.7710e-09	2.9663e-09	1.3356e-08	0.0208	0.0218	0.0402	0.1165
76	0	1.4436e-08	3.4705e-09	1.0472e-09	4.7057e-09	-0.2015	-0.1970	-0.1388	0.0625
77	0	1.7469e-08	4.2187e-09	1.2723e-09	5.7068e-09	0.1405	0.1448	0.1775	0.2156
78	0	1.2665e-08	3.0640e-09	9.2536e-10	4.1531e-09	-0.5740	-0.5521	-0.4080	-0.0698
79	0	3.2731e-08	8.1089e-09	2.4470e-09	1.0854e-08	-0.0591	-0.0544	-0.0118	0.1204
80	0	4.1218e-08	1.0004e-08	3.0317e-09	1.3608e-08	0.9007	0.8601	0.6134	0.2574
81	0	2.0044e-07	4.6478e-08	1.4171e-08	6.5670e-08	0.5438	0.5429	0.5130	0.3221
82	0	2.7913e-08	6.6845e-09	2.0163e-09	9.0646e-09	0.9911	0.9775	0.8581	0.4450
83	0	7.0003e-07	1.2417e-07	3.5835e-08	1.9306e-07	0.1637	0.1718	0.2212	0.2661
84	0	1.7102e-08	4.1675e-09	1.2593e-09	5.6317e-09	-1.1147	-1.0809	-0.8509	-0.2444
85	0	2.0887e-08	5.0443e-09	1.5253e-09	6.8594e-09	-0.7675	-0.7382	-0.5200	-0.0413
86	0	1.4323e-08	3.4808e-09	1.0491e-09	4.7045e-09	0.0517	0.0521	0.0530	0.1017
87	0	3.7620e-08	9.1007e-09	2.7480e-09	1.2319e-08	0.7071	0.7022	0.6533	0.3842
88	0	1.7779e-08	4.2952e-09	1.2997e-09	5.8292e-09	-0.1599	-0.1622	-0.1550	0.0079
89	0	3.9668e-08	9.6895e-09	2.9360e-09	1.3132e-08	-0.0305	-0.0404	-0.0703	0.0523
90	0	3.9154e-08	9.4410e-09	2.8509e-09	1.2799e-08	0.3904	0.3815	0.3381	0.2731
91	0	1.7640e-08	4.2394e-09	1.2821e-09	5.7607e-09	0.0240	0.0226	0.0353	0.1303
92	0	2.0833e-08	5.0597e-09	1.5285e-09	6.8463e-09	0.3514	0.3280	0.1954	0.0792
93	0	8.7248e-08	2.1327e-08	6.4791e-09	2.8877e-08	0.6815	0.6698	0.5851	0.3524
94	0	1.6808e-07	3.9255e-08	1.1951e-08	5.4848e-08	0.8338	0.8289	0.7443	0.3778
95	0	2.9151e-08	7.0695e-09	2.1382e-09	9.5953e-09	0.6605	0.6422	0.5295	0.2894
96	0	1.7789e-07	4.1199e-08	1.2594e-08	5.9480e-08	0.3004	0.2944	0.2549	0.2019
97	0	4.2326e-08	1.0255e-08	3.1042e-09	1.3936e-08	0.4118	0.4202	0.4372	0.2999
98	0	9.1224e-07	1.7460e-07	4.7521e-08	2.7868e-07	0.7233	0.7025	0.5742	0.3062
99	0	4.3751e-08	1.0592e-08	3.2122e-09	1.4496e-08	-0.2966	-0.2931	-0.2451	0.0134
100	0	2.1231e-08	5.1373e-09	1.5504e-09	6.9459e-09	0.8029	0.7774	0.6050	0.2759
101	0	2.7935e-08	6.7510e-09	2.0464e-09	9.1761e-09	-0.3212	-0.2965	-0.1309	0.1448
102	0	3.7096e-08	9.1711e-09	2.7750e-09	1.2293e-08	0.2753	0.2595	0.1631	0.0778
103	0	2.0884e-08	5.0613e-09	1.5256e-09	6.8478e-09	-0.3094	-0.2774	-0.0960	0.1215
104	0	2.5110e-08	6.0247e-09	1.8224e-09	8.2009e-09	0.1875	0.1830	0.1632	0.1645
105	0	1.5328e-08	3.6899e-09	1.1111e-09	4.9954e-09	-0.3619	-0.3322	-0.1403	0.1311
106	0	1.8413e-08	4.3793e-09	1.3219e-09	5.9566e-09	0.4084	0.4187	0.4461	0.3058
107	0	3.0686e-08	7.5135e-09	2.2692e-09	1.0138e-08	0.4653	0.4444	0.3311	0.2081
108	0	2.8227e-08	6.7677e-09	2.0493e-09	9.2244e-09	-0.1157	-0.1036	-0.0331	0.0998
109	0	1.0099e-07	2.3859e-08	7.2925e-09	3.3878e-08	0.4315	0.4196	0.3391	0.2076
110	0	1.1130e-08	2.6669e-09	8.0508e-10	3.6211e-09	0.4779	0.4644	0.3610	0.1583
111	0	1.6264e-08	3.9417e-09	1.1906e-09	5.3300e-09	0.2036	0.1957	0.1454	0.1203
112	0	3.0790e-07	6.5156e-08	2.0034e-08	9.9590e-08	0.4699	0.4559	0.3679	0.2241
113	0	5.5414e-08	1.3546e-08	4.1118e-09	1.8462e-08	0.5338	0.5242	0.4542	0.2762
114	0	3.2952e-08	7.9243e-09	2.3991e-09	1.0821e-08	0.3014	0.2942	0.2522	0.1905
115	0	4.0591e-08	9.6770e-09	2.9263e-09	1.3240e-08	0.2227	0.2318	0.2761	0.2631
116	0	1.8235e-08	4.4486e-09	1.3441e-09	6.0166e-09	-0.3346	-0.3080	-0.1691	0.0425

(b) Elements $59 - 116$ of the optimal point.

Figure 15: Comparison of elements $1 - 116$ of $\mathbf{x}_s$ with the optimal points for various λ.

	Xs	x1, λ=0.01	x1, λ=0.1	x1, λ=1	x1, λ=10	x2, λ=0.01	x2, λ=0.1	x2, λ=1	x2, λ=10
117	0	3.1764e-07	6.7521e-08	2.0787e-08	1.0111e-07	0.4148	0.4111	0.3929	0.2966
118	0	1.4013e-07	3.3740e-08	1.0306e-08	4.6806e-08	1.1217	1.0962	0.9057	0.4304
119	0	1.9729e-08	4.8095e-09	1.4571e-09	6.5251e-09	-0.9989	-0.9755	-0.7895	-0.2101
120	0	1.1180e-07	2.7457e-08	8.3188e-09	3.7083e-08	0.6015	0.5933	0.5371	0.3728
121	0	4.5201e-08	1.1061e-08	3.3445e-09	1.4913e-08	0.0636	0.0777	0.1545	0.2023
122	0	2.6545e-08	6.4343e-09	1.9451e-09	8.7236e-09	0.0523	0.0489	0.0500	0.1510
123	0	1.4021e-08	3.3893e-09	1.0243e-09	4.5964e-09	-0.7085	-0.6934	-0.5687	-0.1434
124	0	2.0782e-08	5.0033e-09	1.5167e-09	6.8317e-09	-0.5067	-0.4800	-0.3098	0.0013
125	0	5.5274e-08	1.3327e-08	4.0388e-09	1.8214e-08	0.3427	0.3383	0.3263	0.2940
126	0	1.8252e-08	4.4231e-09	1.3393e-09	6.0058e-09	-0.4556	-0.4417	-0.3421	-0.0606
127	0	4.9026e-08	1.1985e-08	3.6209e-09	1.6180e-08	1.3209	1.2786	1.0174	0.4954
128	0	1.5732e-08	3.7757e-09	1.1387e-09	5.1223e-09	0.4952	0.4813	0.3954	0.2423

Figure 16: Comparison of elements 117-128 of $\mathbf{x}_s$ with the optimal points for various λ.

	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$	$\lambda = 10$
$f_1\left(\mathbf{x}_{1(opt)}\right)$	0.2362	2.3617	23.5613	230.0178
$f_2\left(\mathbf{x}_{2(opt)}\right)$	0.5575	5.4330	44.2540	192.1869

Table 1: Optimal values of f_1, f_2, for various λ.

In the case of the optimal points of f_1, we observe that they are very close to $\mathbf{x}_s$. They successfully track the non-zero elements of $\mathbf{x}_s$, and the rest of their elements are extremely close to 0, such as in $\mathbf{x}_s$. On the other hand, the elements of the optimal points of f_2 do not differ too much from each other. In general, the optimal points are further from $\mathbf{x}_s$. What is more, while λ increases that assimilation between the elements of the optimal points increases. Those observations justify the reason for which λ is sometimes called 'regularization parameter'.

Furthermore, in what regards the optimal values of f_1, f_2, they increase while λ increases.

Question 6.b. On this section we will solve the following optimization problem:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f_1(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{19}$$

with the use of ISTA & FISTA algorithms. On this problem, we have set $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ and $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$. Moreover, f is a proper closed and convex function and $\text{dom}(f) \equiv \mathbb{R}^n$ is a convex set. Also, g is proper and closed, with $\text{dom}(f) \subseteq \text{int}(\text{dom}(g))$.

In what regards the smoothness of f, it can be written as:

$$f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \frac{1}{2}\mathbf{x}^T\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x} + \left(\mathbf{b}^T\mathbf{A}\right)\mathbf{x} + \mathbf{b}^T\mathbf{b}$$

From example 5.2 of [1] it occurs that f is L_f-smooth, with $L_f = \|\mathbf{A}^T\mathbf{A}\|_{2,2} = \lambda^{max}_{(\mathbf{A}^T\mathbf{A})}$.

The ISTA algorithm is a special case of the proximal gradient method. More in detail, when the proximal gradient algorithm is applied in problems with the form of (19), then the proximal operator is simplified to the soft thresholding operator and it is known as the 'Iterative Shrinkage-Thresholding Algorithm' (ISTA).

The proximal gradient algorithm for problem (19) can be seen below.

Algorithm 5: Proximal Gradient algorithm.

Choose a random $\mathbf{x}_0 \in \text{int}(\text{dom}(f))$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$
  $\mathbf{x}_{k+1} \leftarrow \text{prox}_{t_k g}\left(\mathbf{x}_k - t_k \nabla f(\mathbf{x}_k)\right)$
  $k \leftarrow k + 1$
end

- $\text{int}(\text{dom}(f)) \equiv \mathbb{R}^n$.
- From example 6.8 of [1] we know that, if $\lambda > 0$:

$$\text{prox}_{g=\lambda\|.\|_1}(\mathbf{x}) = \mathcal{T}_\lambda(\mathbf{x}), \text{ where } \mathcal{T}_\lambda(\mathbf{x}) = [\mathcal{T}_\lambda(x_i)]_{i=1}^n = \begin{cases} x_i - \lambda, & \text{if } x_i \geq \lambda \\ 0, & \text{if } |x_i| < \lambda \\ x_i + \lambda, & \text{if } x_i \leq -\lambda \end{cases}$$

- We choose a constant stepsize t_k such that $t_k = \frac{1}{L_k} = \frac{1}{L_f} = \frac{1}{\lambda^{max}_{(\mathbf{A}^T\mathbf{A})}}$ (which will make the algorithm to converge, according to theorems 10.29, 10.34 in [1]).
- $\nabla f(\mathbf{x}) = \nabla\left(\frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2\right) = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$
- As termination criterion for the iterations we will set the following (for $c = 1.01$):

$$f_1(\mathbf{x}_k) \leq c * f_{1(opt)}$$

As a result, algorithm 5 turns into ISTA and is shown at algorithm 6. Moreover, algorithm 7 demonstrates an accelerated variant of ISTA, named FISTA. Both algorithms are also implemented in MATLAB and their results are presented afterwards.

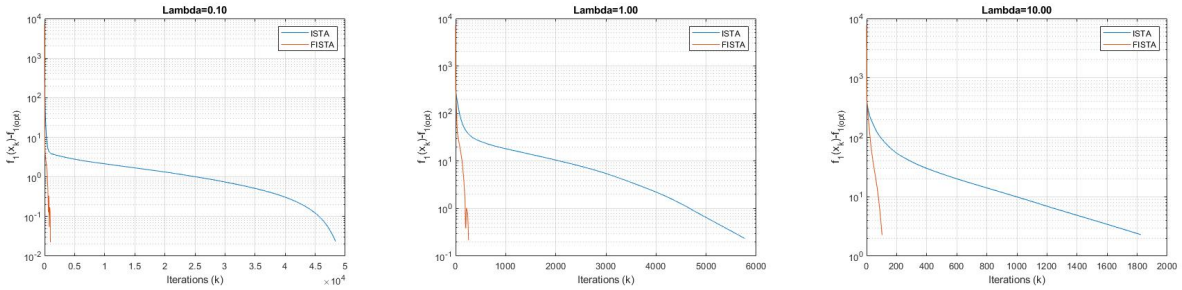
Algorithm 6: ISTA algorithm that solves (19).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$
$k \leftarrow 0$
while $f_1(\mathbf{x}_k) > c * f_{1(opt)}$ **do**
  $\mathbf{x}_{k+1} \leftarrow \mathcal{T}_{\frac{\lambda}{L_f}}\left(\mathbf{x}_k - \frac{1}{L_f}\nabla f(\mathbf{x}_k)\right)$
  $k \leftarrow k + 1$
end

Algorithm 7: FISTA algorithm that solves (19).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$
$\mathbf{y}_0 \leftarrow \mathbf{x}_0$, $t_0 \leftarrow 1$
$k \leftarrow 0$
while $f_1(\mathbf{x}_k) > c * f_{1(opt)}$ **do**
  $\mathbf{x}_{k+1} \leftarrow \mathcal{T}_{\frac{\lambda}{L_f}} \left(\mathbf{y}_k - \frac{1}{L_f} \nabla f(\mathbf{y}_k) \right)$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}} (\mathbf{x}_{k+1} - \mathbf{x}_k)$
  $k \leftarrow k + 1$
end

Figures 17a, 17b, 17c demonstrate $f_1(\mathbf{x}_k) - f_{1(opt)}$ for all three algorithms, and for various values of λ. We can easily see that FISTA has remarkably better performance than ISTA. Another important observation is the speed of convergence versus λ, which is decreased while λ goes closer to 0.



(a) $f_1(\mathbf{x}_k) - f_{1(opt)}, \lambda = 0.1$. (b) $f_1(\mathbf{x}_k) - f_{1(opt)}, \lambda = 1$. (c) $f_1(\mathbf{x}_k) - f_{1(opt)}, \lambda = 10$.

Figure 17: ISTA & FISTA algorithms for various λ.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex06.m`.

Exercise 7

Questions 7.a, 7.b. Initially, we set the dimensions of the problem as $n = 15$ and $m = 20$. Moreover, we construct 2 noise vectors. Both are Gaussian, with expected value equal to 0. The first one's standard deviation is 0.2 (variance 0.04) and corresponds to the case of weak noise, while the second one's standard deviation is 2.5 (variance 6.25) and corresponds to the case of strong noise.

Then, we draw a random matrix $\mathbf{A}$ from the Uniform distribution and generate a piecewise constant vector $\mathbf{x}_{pwc} = [8,8,8,8,9,9,9,1,1,1,1,1,10,10,10]^T$. Finally, we construct vector $\mathbf{b}$ as described in equation (10).

Afterwards, we try to recover $\mathbf{x}_{pwc}$ as the solution of the following optimization problems:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{20}$$

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad F(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \|\mathbf{D}\mathbf{x}\|_1 \tag{21}$$

where $\mathbf{D} \in \mathbb{R}^{(n-1)\text{x}n}$.

Notice that an efficient implementation of $\mathbf{D}\mathbf{x}$ is:

$$\mathbf{D}\mathbf{x} = \mathbf{x}[1 : n-1] - \mathbf{x}[2 : n]$$

If we utilize 'CVX' software, we get the optimal points $\mathbf{x}_f$, $\mathbf{x}_F$ of problems (20), (21) respectively. Both the case of weak and strong noise were tested. The various optimal points that were produced can be compared against each other in fig. 18.

	$\mathbf{x}_{\text{pwc}}$	$\mathbf{x}_{\text{f-WEAK}}$	$\mathbf{x}_{\text{f-STRONG}}$	$\mathbf{x}_{\text{F-WEAK}}$	$\mathbf{x}_{\text{F-STRONG}}$
1	8	7.8071	9.7615	8.0330	9.5780
2	8	7.7116	6.8711	8.0330	9.5780
3	8	8.4690	11.7507	8.0330	9.5780
4	8	8.9884	18.8482	8.0330	11.4820
5	9	9.4606	8.6566	8.6385	9.6208
6	9	9.3389	12.5307	8.6385	9.6208
7	9	8.6989	7.2064	8.6385	7.8977
8	1	0.0782	-1.8695	1.1981	1.0705
9	1	1.0095	0.9851	1.1981	-0.1708
10	1	1.6477	-0.7426	1.1981	-0.1708
11	1	0.2644	-1.9590	1.1981	1.0707
12	1	1.0608	-0.0922	1.1981	1.0707
13	10	9.0064	4.9278	9.8752	7.4906
14	10	10.2375	7.1680	10.0082	7.4906
15	10	10.0213	9.8787	10.0082	9.3336

Figure 18: Efforts to recover $\mathbf{x}_{pwc}$ in various noise conditions.

We observe that, while noise gets stronger, the optimal points of functions f, F approximate less the original $\mathbf{x}_{pwc}$. Moreover, the elements of $\mathbf{x}_F$ are significantly more stable (i.e. similar to each other) compared to $\mathbf{x}_f$ and they approximate better the original piecewise vector $\mathbf{x}_{pwc}$.

Question 7.c. As shown before, problem (21) approximates $\mathbf{x}_{pwc}$ better than problem (20). On this section we will try to solve it with S-FISTA algorithm, instead of 'CVX'. More specifically, it can be re-writen as:

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x}) + g(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \|\mathbf{D}\mathbf{x}\|_1 \tag{22}$$

where $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$, $h(\mathbf{x}) = \|\mathbf{D}\mathbf{x}\|_1$ and $g(\mathbf{x}) = 0$. First of all, we have to verify that f, h, g satisfy the assumptions required by the S-FISTA algorithm:

- $f : \mathbb{R}^n \to \mathbb{R}$ is L_f-smooth, with $L_f = \|\mathbf{A}^T\mathbf{A}\|_{2,2} = \lambda^{max}_{(\mathbf{A}^T\mathbf{A})}$ (proved in section 6.*b*).
- $h : \mathbb{R}^n \to \mathbb{R}$ can be written as $h(\mathbf{x}) = q(\mathbf{D}\mathbf{x})$, where $q(\mathbf{x}) = \|\mathbf{x}\|_1$.

  From example 10.54 in [1] we know that q is (α, β)-smoothable, where $q_\mu(\mathbf{x}) = M_q^\mu(\mathbf{x})$ is a $\frac{1}{\mu}$-smooth ($\mu > 0$) approximation of q with parameters $(\alpha_q, \beta_q) = (1, n/2)$. Notice that n refers to the dimension of $\mathbf{x}$, while M_q^μ denotes the Moreau envelope of q, i.e.

  $$M_q^\mu(\mathbf{x}) = \min_{\mathbf{u}\in\mathbb{R}^n}\left\{q(\mathbf{u}) + \frac{1}{2\mu}\|\mathbf{x}-\mathbf{u}\|^2\right\}$$

  Moreover, in theorem 10.46 of [1] it has been proven that $q_\mu(\mathbf{D}\mathbf{x})$ is a $\frac{1}{\mu}$-smooth approximation of $h(\mathbf{x}) = q(\mathbf{D}\mathbf{x})$ with parameters $(\alpha_h, \beta_h) = (\alpha_q\|\mathbf{D}\|_{2,2}^2, \beta_q) = \left(\|\mathbf{D}\|_{2,2}^2, n/2\right)$ (notice that $\mathbf{D}\mathbf{x}$ is a vector of length n).

  All in all, we proved that h is (α_h, β_h)-smoothable, with $(\alpha_h, \beta_h) = \left(\|\mathbf{D}\|_{2,2}^2, n/2\right)$. Its $\frac{1}{\mu}$-smooth approximation is $h_\mu(\mathbf{x}) = M_q^\mu(\mathbf{D}\mathbf{x})$.
- $g(\mathbf{x}) = 0$, with $\text{dom}(g) = \mathbb{R}^n$, is proper closed and convex.
- The level sets of function F are bounded, i.e. $\forall\delta > 0,\ \exists\, R_\delta > 0$ such that $\|\mathbf{x}\| \le R_\delta,\ \forall\,\mathbf{x} : F(\mathbf{x}) \le \delta$.

The S-FISTA algorithm is the following:

Algorithm 8: S-FISTA algorithm.

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$ and a value for parameter $\mu > 0$
$\mathbf{y}_0 \leftarrow \mathbf{x}_0$, $t_0 \leftarrow 1$
Construct h_μ to be a $\frac{1}{\mu}$-smooth approximation of h with parameters (α, β)
$F_\mu(\mathbf{x}) = f(\mathbf{x}) + h_\mu(\mathbf{x})$, $\bar{L} \leftarrow L_f + \frac{\alpha}{\mu}$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  $\mathbf{x}_{k+1} \leftarrow \text{prox}_{\frac{g}{\bar{L}}}\left(\mathbf{y}_k - \frac{1}{\bar{L}}\nabla F_\mu(\mathbf{y}_k)\right)$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}_k)$
  $k \leftarrow k + 1$
end

- We smooth the cost function F, and we get:

$$F_\mu(\mathbf{x}) = f(\mathbf{x}) + h_\mu(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + M_q^\mu(\mathbf{D}\mathbf{x})$$

- $\bar{L} = L_f + \frac{\alpha_h}{\mu} = L_f + \frac{\|\mathbf{D}\|_{2,2}^2}{\mu}$

- As a criterion to terminate the iterations we will use:

$$F(\mathbf{x}) - F_{opt} \leq \epsilon$$

  In order to get a solution located at a distance $\leq \epsilon$ from the optimal one (and according to theorem 10.57 in [1]), we have to set the smoothing parameter μ as follows:

$$\mu = \sqrt{\frac{\alpha_h}{\beta_h}} \frac{\epsilon}{\sqrt{\alpha_h \beta_h} + \sqrt{\alpha_h \beta_h + L_f \epsilon}}$$

- For any $\mathbf{z} \in \mathbb{R}^n$:

$$\text{prox}_{\frac{g}{\bar{L}}}(\mathbf{z}) = \text{prox}_0(\mathbf{z}) = \mathbf{z}$$

- The gradient of F_μ can be computed as below:

$$\begin{aligned}
\nabla F_\mu(\mathbf{x}) &= \nabla f(\mathbf{x}) + \mathbf{D}^T \nabla M_q^\mu(\mathbf{D}\mathbf{x}) \overset{\text{see section } A6.b \text{ \& theor. 6.60 in [1]}}{=\!=\!=} \\
&= \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{1}{\mu}\mathbf{D}^T(\mathbf{D}\mathbf{x} - \text{prox}_{\mu q}(\mathbf{D}\mathbf{x})) \overset{\text{see example 6.8 in [1]}}{=\!=\!=} \\
&= \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{1}{\mu}\mathbf{D}^T(\mathbf{D}\mathbf{x} - \mathcal{T}_\mu(\mathbf{D}\mathbf{x}))
\end{aligned}$$

  where, for a description of $\mathcal{T}_\mu$, see section 6.*b*.

Consequently, the S-FISTA algorithm (adapted for problem (22)) turns into the following:

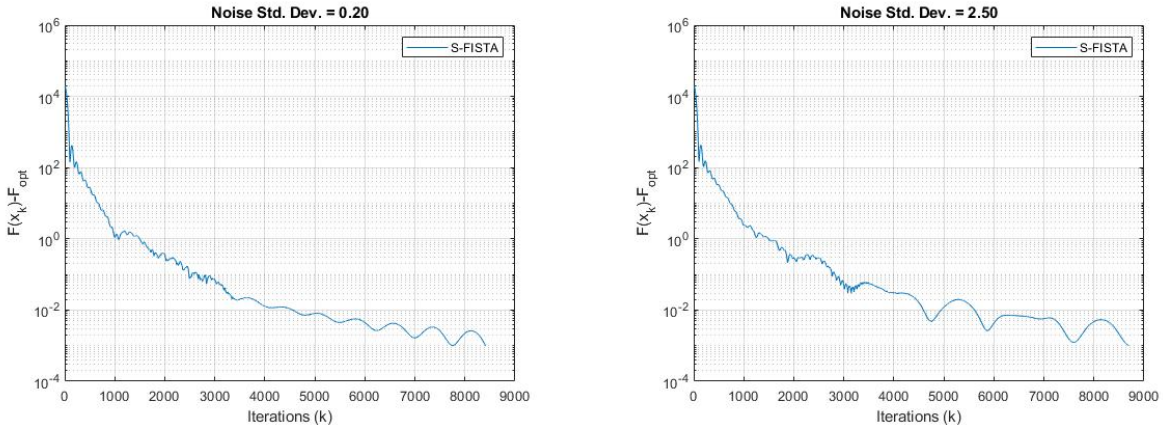
Algorithm 9: S-FISTA algorithm that solves (22).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$ and a value for stop criterion $\epsilon > 0$
$(\alpha_h, \beta_h) \leftarrow \left(\|\mathbf{D}\|_{2,2}^2, n/2\right)$, $\mu \leftarrow \sqrt{\frac{\alpha_h}{\beta_h}} \frac{\epsilon}{\sqrt{\alpha_h \beta_h} + \sqrt{\alpha_h \beta_h + \|\mathbf{A}^T\mathbf{A}\|_{2,2}\epsilon}}$
$\mathbf{y}_0 \leftarrow \mathbf{x}_0$, $t_0 \leftarrow 1$, $\bar{L} \leftarrow \|\mathbf{A}^T\mathbf{A}\|_{2,2} + \frac{\alpha_h}{\mu}$
$k \leftarrow 0$
while $F(\mathbf{x}_k) - F_{opt} > \epsilon$ **do**
  $\mathbf{x}_{k+1} \leftarrow \mathbf{y}_k - \frac{1}{\bar{L}}\left(\mathbf{A}^T(\mathbf{A}\mathbf{y}_k - \mathbf{b}) + \frac{1}{\mu}\mathbf{D}^T(\mathbf{D}\mathbf{y}_k - \mathcal{T}_\mu(\mathbf{D}\mathbf{y}_k))\right)$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}_k)$
  $k \leftarrow k + 1$
end

Products $\mathbf{Dx}$, $\mathbf{D}^T\mathbf{y}$ are implemented efficiently as follows:

$$\mathbf{Dx} = \mathbf{x}[1 : n-1] - \mathbf{x}[2 : n]$$
$$\mathbf{D}^T\mathbf{y} = \begin{bmatrix}\mathbf{y}\\ 0\end{bmatrix} - \begin{bmatrix}0\\ \mathbf{y}\end{bmatrix}$$

Figures 19a, 19b demonstrate the quantity $F(\mathbf{x}_k) - F_{opt}$ for both the case of weak and strong noise, as they were described in the previous section. The initial point $\mathbf{x}_0$ was drawn randomly, and parameter ϵ was set to 10^{-3}. Moreover, parameter p for the construction of matrix $\mathbf{D}$ was set equal to 1. We observe that in both the weak and the strong noise case, the number of iterations is about the same.



(a) Weak noise ($\sigma = 0.2$). (b) Strong noise ($\sigma = 2.5$).

Figure 19: S-FISTA algorithm for $m = 20$, $n = 15$ and various noise amounts.

Question 7.d. On this section we will solve problem (22) again, but this time we will utilize the subgradient descent method, as described in section 8.2.1 of [1]. We will test it with the Polyak stepsize, which is the optimal one.

The assumptions required by this algorithm are satisfied, since F is proper closed and convex function, and $\mathbb{R}^n$ is a non-empty closed and convex set. Moreover, it is true that $\mathbf{P}_{\mathbb{R}^n}(\mathbf{x}) = \mathbf{x}$, $\forall \mathbf{x} \in \mathbb{R}^n$. As a result, the algorithm turns into algorithm 1 again, as presented in section 4.*a*, *A*4.*b*.

- As a subgradient of $h(\mathbf{x}) = \|\mathbf{Dx}\|_1$ we will use the following weak result, which can be easily extracted from example 3.44 of [1]:

  $$h'(\mathbf{x}) = \mathbf{D}^T \mathrm{sgn}(\mathbf{Dx})$$

  The subdifferential set of $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|_2^2$ can be easily extracted if we combine the results of examples 3.45 and 3.48 of [1]:

  $$\partial f(\mathbf{x}) = \begin{cases}\|\mathbf{Ax}-\mathbf{b}\|_2 \frac{\mathbf{A}^T(\mathbf{Ax}-\mathbf{b})}{\|\mathbf{Ax}-\mathbf{b}\|_2} \\ \|\mathbf{Ax}-\mathbf{b}\|_2 B_{\|.\|_2}[0,1]\end{cases} = \begin{cases}\mathbf{A}^T(\mathbf{Ax}-\mathbf{b}), & \text{if } \mathbf{Ax}-\mathbf{b} \neq \mathbf{0} \\ \|\mathbf{Ax}-\mathbf{b}\|_2 B_{\|.\|_2}[0,1], & \text{if } \mathbf{Ax}-\mathbf{b} = \mathbf{0}\end{cases}$$

Since $\mathbf{0} \in B_{\|.\|_2}[0,1]$ where $B_{\|.\|_2}[0,1]$ is the ℓ_2-unit ball set, we can use the following result as a subgradient of f:

$$f'(\mathbf{x}) = \begin{cases} \mathbf{A}^T(\mathbf{Ax}-\mathbf{b}), & \text{if } \mathbf{Ax}-\mathbf{b} \neq \mathbf{0} \\ \mathbf{0}, & \text{if } \mathbf{Ax}-\mathbf{b} = \mathbf{0} \end{cases} = \mathbf{A}^T(\mathbf{Ax}-\mathbf{b})$$

Finally, from theorem 3.40 of [1] we get a subgradient for the cost function F:

$$F'(\mathbf{x}) = f'(\mathbf{x}) + h'(\mathbf{x}) = \mathbf{A}^T(\mathbf{Ax}-\mathbf{b}) + \mathbf{D}^T\text{sgn}(\mathbf{Dx})$$

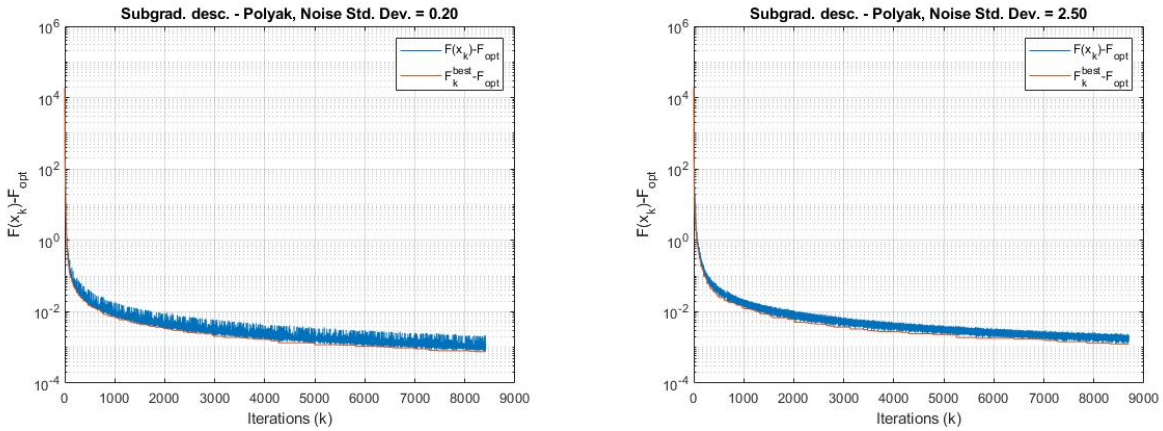
- The Polyak step is:

$$t_k = \begin{cases} \frac{F(\mathbf{x}_k)-F_{opt}}{\|F'(\mathbf{x}_k)\|^2}, & F'(\mathbf{x}_k) \neq 0 \\ 1, & F'(\mathbf{x}_k) = 0 \end{cases}$$

where F_{opt} is the solution of the optimization problem (22), which was estimated by 'CVX'.

Products $\mathbf{Dx}$, $\mathbf{D}^T\mathbf{y}$ are implemented efficiently as described in section 7.*b*.

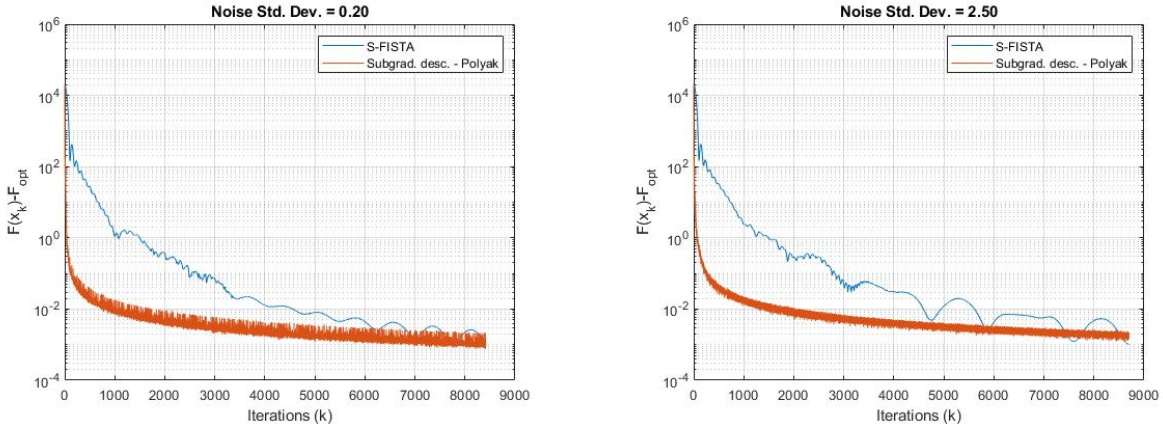
Figures 20a, 20b show the performance of the subgradient algorithm with Polyak step for weak and strong noise conditions, respectively. The parameters are the same with the previous section (random $\mathbf{x}_0$, $p = 1$), while the number of iterations was set to be equal with the respective iterations that were required at the previous step. Each plot demonstrates the actual value of the algorithm at every iteration $F(\mathbf{x}_k) - F_{opt}$ (blue), as well as the best value achieved up to the k-th iteration $F_k^{best} - F_{opt}$ (red).



(a) Weak noise ($\sigma = 0.2$). (b) Strong noise ($\sigma = 2.5$).

Figure 20: Projected subgradient descent algorithm with Polyak step.

Question 7.e. On this section we combine the plots from 4.*c* (blue) and 4.*d* (red). We observe that the optimal case of the projected subgradient algorithm (Polyak stepsize) has best performance when the iterations are few. However, while iterations increase, the precision and rate of convergence of S-FISTA become equivalent with the subgradient algorithm's.



(a) Weak noise ($\sigma = 0.2$). (b) Strong noise ($\sigma = 2.5$).

Figure 21: Comparison of S-FISTA with subgradient algorithm.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex07.m`.

Exercise 8

Question 8.a. On this exercise we have to solve the following optimization problem with various techniques ($\lambda > 0$):

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 + \lambda \|\mathbf{x}\|_1 \tag{23}$$

On this section we will attempt to solve it with the projected subgradient descent algorithm, with both the Polyak and the dynamic stepsize, as described in section 8.2.1 of [1]. The assumptions required by this algorithm are satisfied, since F is proper closed and convex function, and $\mathbb{R}^n$ is a non-empty closed and convex set. Moreover, it is true that $\mathbf{P}_{\mathbb{R}^n}(\mathbf{x}) = \mathbf{x}, \ \forall \mathbf{x} \in \mathbb{R}^n$. As a result, the algorithm turns into algorithm 1 again, as presented in section 4.*a*, *A*4.*b*.

- As a subgradient for $f(\mathbf{x}), g(\mathbf{x})$ we will use the following weak results, which can be easily extracted from example 3.44 of [1]:

  $$f'(\mathbf{x}) = \mathbf{A}^T \text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b})$$
  $$g'(\mathbf{x}) = \lambda \text{sgn}(\mathbf{x})$$

  Finally, from theorem 3.40 of [1] we get a subgradient for the cost function F:

  $$F'(\mathbf{x}) = f'(\mathbf{x}) + g'(\mathbf{x}) = \mathbf{A}^T \text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \lambda \text{sgn}(\mathbf{x})$$

- The Polyak step is:

  $$t_k = \begin{cases} \frac{F(\mathbf{x}_k) - F_{opt}}{\|F'(\mathbf{x}_k)\|^2}, & F'(\mathbf{x}_k) \neq 0 \\ 1, & F'(\mathbf{x}_k) = 0 \end{cases}$$

  where F_{opt} is the solution of the optimization problem (23), which was estimated by 'CVX'.

- As dynamic stepsize the following (theorem 8.28 at [1]) will be used:

  $$t_k = \begin{cases} \frac{1}{\|F'(\mathbf{x}_k)\| \sqrt{k+1}}, & \text{if } \|F'(\mathbf{x}_k)\| \neq 0 \\ \frac{1}{L_F}, & \text{if } \|F'(\mathbf{x}_k)\| = 0 \end{cases}$$

  where $L_F \geq \|\mathbf{w}\|_2, \forall \mathbf{w} \in \partial F(\mathbf{x})$.

  Let $h(\mathbf{x}) = \|\mathbf{x}\|_1$. From example 3.41 in [1] it can be shown that $\|\partial h(\mathbf{x})\|_2 \leq \sqrt{n}$, for any $\mathbf{x} \in \mathbb{R}^n$. As a result, we get that:

  $$\begin{aligned} &f(\mathbf{x}) = h(\mathbf{A}\mathbf{x} - \mathbf{b}) \Rightarrow \partial f(\mathbf{x}) = \mathbf{A}^T \partial h(\mathbf{A}\mathbf{x} - \mathbf{b}) \Rightarrow \\ \Rightarrow &\|\partial f(\mathbf{x})\|_2 = \|\mathbf{A}^T \partial h(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2 \leq \|\mathbf{A}^T\|_{2,2} \|\partial h(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2 \leq \|\mathbf{A}^T\|_{2,2} \sqrt{m} \\ &g(\mathbf{x}) = \lambda h(\mathbf{x}) \Rightarrow \partial g(\mathbf{x}) = \lambda \partial h(\mathbf{x}) \Rightarrow \\ \Rightarrow &\|\partial g(\mathbf{x})\|_2 = \|\lambda \partial h(\mathbf{x})\|_2 \overset{\lambda > 0}{=\!=} \lambda \|\partial h(\mathbf{x})\|_2 \leq \lambda \sqrt{n} \end{aligned}$$

So, from the triangular inequality we get that:

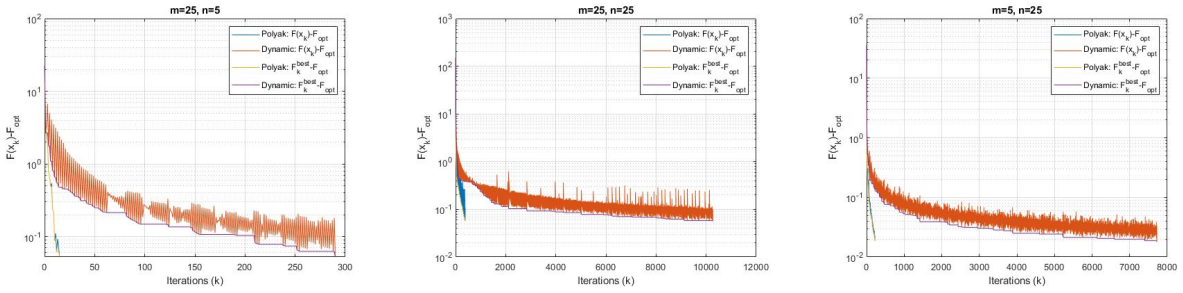
$$\|\partial F(\mathbf{x})\|_2 = \|\partial f(\mathbf{x}) + \partial g(\mathbf{x})\|_2 \leq \|\partial f(\mathbf{x})\|_2 + \|\partial g(\mathbf{x})\|_2 = \|\mathbf{A}^T\|_{2,2}\sqrt{m} + \lambda\sqrt{n}$$

The above result shows that, $\forall \mathbf{w} \in \partial F(\mathbf{x})$, it holds that $\|\mathbf{w}\|_2 \leq \|\mathbf{A}^T\|_{2,2}\sqrt{m} + \lambda\sqrt{n}$. This means that we can set:

$$L_F = \|\mathbf{A}^T\|_{2,2}\sqrt{m} + \lambda\sqrt{n}$$

In figures 22a, 22b, 22c we can see 3 different executions of the algorithms, for (m, n) dimensions $(25, 5)$, $(25, 25)$, $(5, 25)$ respectively. We set $\lambda = 1$ and the termination criterion $c = 1.01$. Moreover, the initial point $\mathbf{x}_0$ as well as matrices $\mathbf{A}$, $\mathbf{b}$ were randomly drawn.

We observe that in all cases the Polyak step performs much better than the dynamic one. Also, when $m >> n$ less iterations are required compared to the other cases.



(a) $m = 25, n = 5$. (b) $m = 25, n = 25$. (c) $m = 5, n = 25$.

Figure 22: Projected subgradient alg. with Polyak & dynamic steps, for various m, n.

Question 8.b. The proximal subgradient technique, which is defined in remark 9.23 of [1], can be also used to solve problem (23). Both f and g are proper closed and convex functions with $\text{dom}(g) \equiv \text{dom}(f)$. Furthermore, in 8.*a* we proved that exists a constant $L_f > 0$ such that $\|f'(\mathbf{x})\| \leq L_f$. As a result, all assumptions of this method are satisfied.

Again, we will use the dynamic stepsize t_k and the subgradient $f'(\mathbf{x})$ that were defined in the previous section. Attention is needed on the fact that we must use the subgradient of function f, and not of F, as previously. Moreover, in section 6.*b* it was proved that the proximal operator prox_g of $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ is equal to the soft thresholding operator $\mathcal{T}_\lambda$. After all, the (adapted for (23)) algorithm turns into the following:

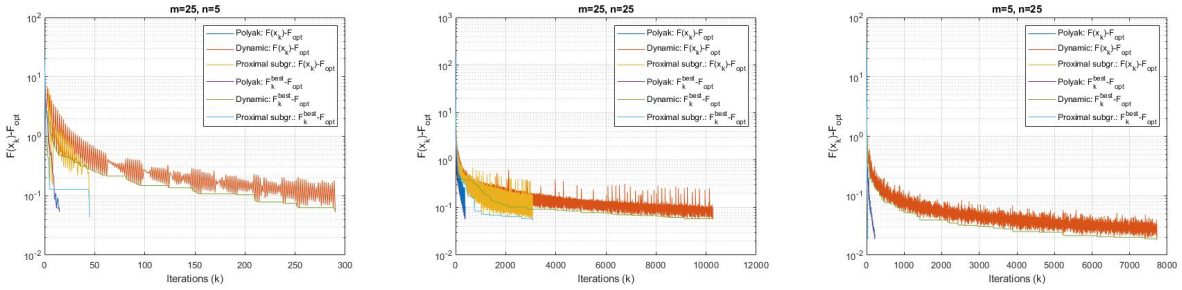
Algorithm 10: Proximal subgradient algorithm for solving (23).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $f'(\mathbf{x}_k) \in \partial f(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathcal{T}_{t_k\lambda}\left(\mathbf{x}_k - t_k f'(\mathbf{x}_k)\right)$
  $k \leftarrow k + 1$
end

In figures 23a, 23b, 23c we combine the plots from the previous section with the respective executions of this section's algorithm, for (m, n) dimensions $(25, 5)$, $(25, 25)$,

$(5, 25)$. Same as previously, we set $\lambda = 1$ and the termination criterion $c = 1.01$. Also, matrices $\mathbf{A}, \mathbf{b}$, as well as the initial point $\mathbf{x}_0$ are the same with before.

The main observation is the same with previously: when $m >> n$, then less iterations are required compared to the other cases. Moreover, it is obvious that the performance of the proximal subgradient method is better than the projected subgradient with dynamic substeps, independently of m, n. Also, we see that the proximal subgradient has an extremely good performance, when $m << n$.



(a) $m = 25, n = 5$. (b) $m = 25, n = 25$. (c) $m = 5, n = 25$.

Figure 23: Proximal subgradient algorithm, for various m, n.

Question 8.c. On this section we will employ S-FISTA to solve problem (23). We will smoothen only f, with λg being kept at its original form, in order to be utilized at the proximal step.

As a first step, we take care so that all requirements of S-FISTA to be satisfied:

- $q(\mathbf{x}) = 0$ is a 0-smooth function.
- f is an (α_f, β_f)-smoothable function. Its $\frac{1}{\mu}$-smooth approximation is $f_\mu(\mathbf{x}) = M_h^\mu(\mathbf{A}\mathbf{x} - \mathbf{b})$, with parameters $(\alpha_f, \beta_f) = \left(\|\mathbf{A}\|_{2,2}^2, m/2\right)$.

  Proof: We define an auxiliary function $h(\mathbf{x}) = \|\mathbf{x}\|_1, \mathbf{x} \in \mathbb{R}^n$. From example 10.54 in [1] we know that $h(\mathbf{x}) = \|\mathbf{x}\|_1$ is (α, β)-smoothable, where $h_\mu(\mathbf{x}) = M_h^\mu(\mathbf{x})$ is a $\frac{1}{\mu}$-smooth $(\mu > 0)$ approximation of h with parameters $(\alpha_h, \beta_h) = (1, n/2)$. Notice that n refers to the dimension of $\mathbf{x}$, while M_h^μ denotes the Moreau envelope of h, i.e.

$$M_h^\mu(\mathbf{x}) = \min_{\mathbf{u} \in \mathbb{R}^n} \left\{ h(\mathbf{u}) + \frac{1}{2\mu} \|\mathbf{x} - \mathbf{u}\|^2 \right\}.$$

  By theorem 10.46 of [1] it has been proven that $h_\mu(\mathbf{A}\mathbf{x} - \mathbf{b})$ is a $\frac{1}{\mu}$-smooth approximation of $f(\mathbf{x}) = h(\mathbf{A}\mathbf{x} - \mathbf{b})$ with parameters $(\alpha_f, \beta_f) = (\alpha_h \|\mathbf{A}\|_{2,2}^2, \beta_h) = \left(\|\mathbf{A}\|_{2,2}^2, m/2\right)$ (notice that $\mathbf{A}\mathbf{x} - \mathbf{b}$ is a vector of length m).

  This means that f is (α_f, β_f)-smoothable, with $(\alpha_f, \beta_f) = \left(\|\mathbf{A}\|_{2,2}^2, m/2\right)$. Its $\frac{1}{\mu}$-smooth approximation is $f_\mu(\mathbf{x}) = M_h^\mu(\mathbf{A}\mathbf{x} - \mathbf{b})$.

- λg is a proper closed and convex function.

The general form of S-FISTA can be seen in section 7.*c* (as well as in 10.8.4 of [1]). Thus, we have to estimate some of the parameters, before we specifically implement it on problem (23).

- We smoothen function f, and we get:

$$f_\mu(\mathbf{x}) = M^{\mu}_{\|.\|_1}(\mathbf{Ax}-\mathbf{b}) \text{ with parameters } (\alpha_f, \beta_f) = \left(\|\mathbf{A}\|^2_{2,2}, m/2\right)$$

- $\bar{L} = 0 + \frac{\alpha_f}{\mu} = \frac{\|\mathbf{A}\|^2_{2,2}}{\mu}$
- As a criterion to terminate the iterations we will use:

$$F(\mathbf{x}) \leq cF_{opt} \Leftrightarrow F(\mathbf{x}) - F_{opt} \leq (c-1)F_{opt} = \epsilon$$

  In order to get a solution located at a distance $\leq \epsilon$ from the optimal one (and according to theorem 10.57 in [1]), we have to set the smoothing parameter μ of function g as follows:

$$\mu = \sqrt{\frac{\alpha_f}{\beta_f}}\frac{\epsilon}{\sqrt{\alpha_f\beta_f}+\sqrt{\alpha_f\beta_f}}$$

- For any $\mathbf{z} \in \mathbb{R}^n$:

$$\text{prox}_{\frac{\lambda g}{\bar{L}}}(\mathbf{z}) = \text{prox}_{\frac{1}{\bar{L}}\|.\|_1}(\mathbf{z}) \overset{\text{see example 6.8 in [1]}}{=\!=\!=} \mathcal{T}_{\frac{\lambda}{\bar{L}}}(\mathbf{z})$$

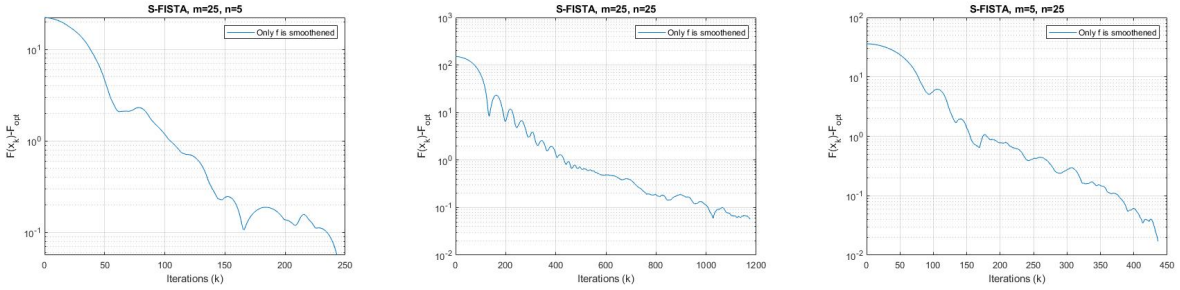
- The gradient of f_μ can be computed as below:

$$\begin{aligned}\nabla f_\mu(\mathbf{x}) &\overset{\text{see theor. 6.60 in [1]}}{=\!=\!=} \frac{1}{\mu}\mathbf{A}^T(\mathbf{Ax}-\mathbf{b}-\text{prox}_{\mu\|.\|_1}(\mathbf{Ax}-\mathbf{b})) \overset{\text{see example 6.8 in [1]}}{=\!=\!=} \\ &= \frac{1}{\mu}\mathbf{A}^T(\mathbf{Ax}-\mathbf{b}-\mathcal{T}_\mu(\mathbf{Ax}-\mathbf{b}))\end{aligned}$$

  where, for a description of $\mathcal{T}_\mu$, see section 6.*b*.

In figures 24a, 24b, 24c we can see 3 different executions of S-FISTA, for (m, n) dimensions $(25, 5)$, $(25, 25)$, $(5, 25)$ respectively. We set $\lambda = 1$ and the termination criterion $c = 1.01$. Moreover, the initial point $\mathbf{x}_0$ as well as matrices $\mathbf{A}$, $\mathbf{b}$ are the same with before.

We observe again that while m decreases and n increases, the number of iterations that is required for convergence increases.



(a) $m = 25, n = 5$. (b) $m = 25, n = 25$. (c) $m = 5, n = 25$.

Figure 24: S-FISTA after smoothening f, for various m, n.

Question 8.d. On this section we will employ S-FISTA again, in order to solve problem (23). However, this time both f and g will be smoothened, so no proximal step is required. Initially, let's check again that all requirements of S-FISTA are satisfied:

- $q(\mathbf{x}) = 0$ is a 0-smooth function.
- $r(\mathbf{x}) = 0$ is a proper closed and convex function.
- We have to prove that $f + g$ is $(\alpha_{f+g}, \beta_{f+g})$-smoothable.

  We define an auxiliary function $h(\mathbf{x}) = \|\mathbf{x}\|_1, \mathbf{x} \in \mathbb{R}^n$. From example 10.54 in [1] we know that $h(\mathbf{x}) = \|\mathbf{x}\|_1$ is (α, β)-smoothable, where $h_\mu(\mathbf{x}) = M_h^\mu(\mathbf{x})$ is a $\frac{1}{\mu}$-smooth $(\mu > 0)$ approximation of h with parameters $(\alpha_h, \beta_h) = (1, n/2)$. Notice that n refers to the dimension of $\mathbf{x}$, while M_h^μ denotes the Moreau envelope of h, i.e.

  $$M_h^\mu(\mathbf{x}) = \min_{\mathbf{u} \in \mathbb{R}^n} \left\{ h(\mathbf{u}) + \frac{1}{2\mu} \|\mathbf{x} - \mathbf{u}\|^2 \right\}.$$

  - By theorem 10.46 of [1] it has been proven that $h_\mu(\mathbf{Ax} - \mathbf{b})$ is a $\frac{1}{\mu}$-smooth approximation of $f(\mathbf{x}) = h(\mathbf{Ax} - \mathbf{b})$ with parameters $(\alpha_f, \beta_f) = (\alpha_h \|\mathbf{A}\|_{2,2}^2, \beta_h) = \left(\|\mathbf{A}\|_{2,2}^2, m/2 \right)$ (notice that $\mathbf{Ax} - \mathbf{b}$ is a vector of length m).

    This means that f is (α_f, β_f)-smoothable, with $(\alpha_f, \beta_f) = \left(\|\mathbf{A}\|_{2,2}^2, m/2 \right)$. Its $\frac{1}{\mu}$-smooth approximation is $f_\mu(\mathbf{x}) = M_h^\mu(\mathbf{Ax} - \mathbf{b})$.
  - By theorem 10.46 of [1] it has been proven that $\lambda h_\mu(\mathbf{x})$ is a $\frac{1}{\mu}$-smooth approximation of $\lambda g(\mathbf{x}) = \lambda h(\mathbf{x})$ with parameters $(\alpha_g, \beta_g) = (\lambda \alpha_h, \lambda \beta_h) = (\lambda, (\lambda n)/2)$.

    As a result, λg is $(\alpha_{\lambda g}, \beta_{\lambda g})$-smoothable, with $(\alpha_{\lambda g}, \beta_{\lambda g}) = (\lambda, (\lambda n)/2)$. Its $\frac{1}{\mu}$-smooth approximation is $\lambda g_\mu(\mathbf{x}) = \lambda M_h^\mu(\mathbf{x})$.

  According to theorem 10.46 in [1], summation preserves smoothability and approximability. Thus, $f_\mu(\mathbf{x}) + \lambda g_\mu(\mathbf{x}) = M_h^\mu(\mathbf{Ax} - \mathbf{b}) + \lambda M_h^\mu(\mathbf{x})$ is a $\frac{1}{\mu}$-smooth approximation of $f + \lambda g$, with $(\alpha_{f+\lambda g}, \beta_{f+\lambda g}) = (\alpha_f + \alpha_{\lambda g}, \beta_f + \beta_{\lambda g}) = (\|\mathbf{A}\|_{2,2}^2 + \lambda, m/2 + (\lambda n)/2)$.

The general form of S-FISTA can be seen in section 7.*c* (as well as in 10.8.4 of [1]). Thus, we have to estimate some of the parameters, before we specifically implement it on problem (23).

- We smoothen function $f + \lambda g$, and we get:

  $$(f + \lambda g)_\mu(\mathbf{x}) = M_{\|\cdot\|_1}^\mu(\mathbf{Ax} - \mathbf{b}) + \lambda M_{\|\cdot\|_1}^\mu(\mathbf{x})$$
  $$\text{with parameters } (\alpha_{f+\lambda g}, \beta_{f+\lambda g}) = (\|\mathbf{A}\|_{2,2}^2 + \lambda, m/2 + (\lambda n)/2)$$

- $\bar{L} = 0 + \frac{\alpha_{f+\lambda g}}{\mu} = \frac{\|\mathbf{A}\|_{2,2}^2 + \lambda}{\mu}$

- As a criterion to terminate the iterations we will use:

$$F(\mathbf{x}) \leq cF_{opt} \Leftrightarrow F(\mathbf{x}) - F_{opt} \leq (c-1)F_{opt} = \epsilon$$

  In order to get a solution located at a distance $\leq \epsilon$ from the optimal one (and according to theorem 10.57 in [1]), we have to set the smoothing parameter μ of function $f + g$ as follows:

$$\mu = \sqrt{\frac{\alpha_{f+\lambda g}}{\beta_{f+\lambda g}}} \frac{\epsilon}{\sqrt{\alpha_{f+\lambda g}\beta_{f+\lambda g}} + \sqrt{\alpha_{f+\lambda g}\beta_{f+\lambda g}}}$$

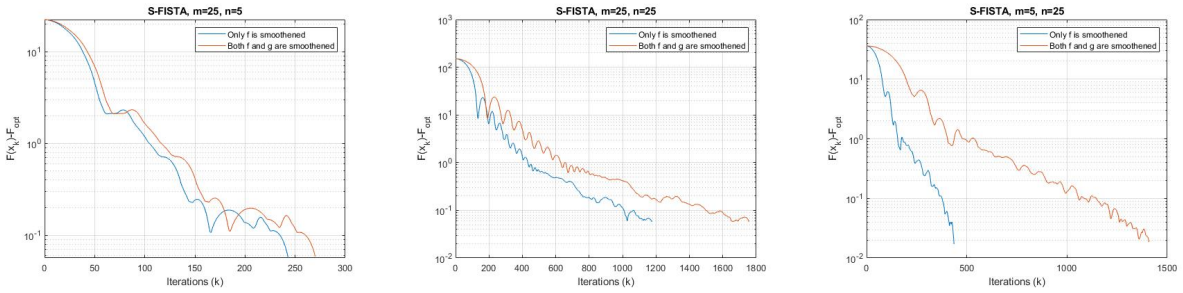
- For any $\mathbf{z} \in \mathbb{R}^n$:

$$\text{prox}_0(\mathbf{z}) = \mathbf{z}$$

- The gradient of the smoothened function F_μ is:

$$\begin{aligned}
\nabla(f+\lambda g)_\mu(\mathbf{x}) =& \nabla f_\mu(\mathbf{x}) + \nabla \lambda g_\mu(\mathbf{x}) \overset{\text{see theor. 6.60 in [1]}}{=\!=} \\
=& \frac{1}{\mu}\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b} - \text{prox}_{\mu\|.\|_1}(\mathbf{A}\mathbf{x} - \mathbf{b})) + \frac{\lambda}{\mu}(\mathbf{x} - \text{prox}_{\mu\|.\|_1}(\mathbf{x})) \overset{\text{see example 6.8 in [1]}}{=\!=} \\
=& \frac{1}{\mu}\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b} - \mathcal{T}_\mu(\mathbf{A}\mathbf{x} - \mathbf{b})) + \frac{\lambda}{\mu}(\mathbf{x} - \mathcal{T}_\mu(\mathbf{x}))
\end{aligned}$$

  where, for a description of $\mathcal{T}_\mu$, see section 6.*b*.



(a) $m = 25, n = 5$. (b) $m = 25, n = 25$. (c) $m = 5, n = 25$.

Figure 25: S-FISTA after smoothening both f and g, for various m, n.

In figures 25a, 25b, 25c we combine the plots from the previous section 8.*c* with the respective executions of this section's S-FISTA algorithm, for (m, n) dimensions $(25, 5)$, $(25, 25)$, $(5, 25)$. Same as previously, we set $\lambda = 1$ and the termination criterion $c = 1.01$. Also, matrices $\mathbf{A}, \mathbf{b}$, as well as the initial point $\mathbf{x}_0$ are the same with before.

Again, when $m << n$ more observations are required than when $m >> n$. Furthermore, when $m >> n$ the performance of S-FISTA is almost identical in the case of smoothening only f (proximal step exists) and of smoothening both f and g (no proximal step). On the contrary, when $n = m$ and $n >> m$ there is a significant difference between the cases. The algorithm with the proximal step is faster than the algorithm without it.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex08.m`.

Exercise 9

Question 9.a. On this exercise we will solve the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & h(\mathbf{x}) = \|\mathbf{x}\|_1 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{24}$$

As a first step we draw $\mathbf{A}$, $\mathbf{b}$ randomly, and solve the problem via 'CVX'. Then, we keep the optimal point $\mathbf{x}_*$ and the optimal value h_{opt} to use them in the next sections.

Question 9.b. From example 10.54 in [1] we know that the $\frac{1}{\mu}$-smooth approximation of h is its Moreau envelope $M_h^\mu(\mathbf{x})$, with parameters $(\alpha_h, \beta_h) = (1, n/2)$. The Moreau envelope has been defined in detail at section 7.*c* of this report.

Also, we can define $f(\mathbf{x}) = 0$, and $g(\mathbf{x}) = \delta_{\mathbf{Ax}=\mathbf{b}} = \begin{cases} 0, & \text{if } \mathbf{Ax} = \mathbf{b} \\ +\infty, & \text{if } \mathbf{Ax} \neq \mathbf{b} \end{cases}$.

It can be easily understood that problem (24) is equivalent with:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad H(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x}) + g(\mathbf{x}) = \|\mathbf{x}\|_1 + \delta_{\mathbf{Ax}=\mathbf{b}} \tag{25}$$

All assumptions required by S-FISTA are satisfied, since:

- f is a 0-smooth function.
- g is a proper closed and convex function.
- h is (α_h, β_h)-smoothable, with h_μ denoting its $\frac{1}{\mu}$-approximation.
- The level sets of H are bounded, i.e. $\forall \delta > 0,\ \exists\, R_\delta > 0$ such that $\|\mathbf{x}\| \leq R_\delta,\ \forall\, \mathbf{x} : H(\mathbf{x}) \leq \delta$.

The general form of S-FISTA can be seen in section 7.*c* (as well as in 10.8.4 of [1]). Thus, we have to estimate some of the parameters, before we specifically implement it on problem (25).

- We smooth the cost function $H - g$, and we get:

$$H_\mu(\mathbf{x}) = f_(\mathbf{x}) + h_\mu(\mathbf{x}) = M_h^\mu(\mathbf{x})$$

- $\bar{L} = 0 + \frac{\alpha_h}{\mu} = \frac{1}{\mu}$
- As a criterion to terminate the iterations we will use:

$$H(\mathbf{x}) - H_{opt} = h(\mathbf{x}) - h_{opt} \leq \epsilon$$

  In order to get a solution located at a distance $\leq \epsilon$ from the optimal one (and according to theorem 10.57 in [1]), we have to set the smoothing parameter μ of function g as follows:

$$\mu = \sqrt{\frac{\alpha_h}{\beta_h}} \frac{\epsilon}{\sqrt{\alpha_h \beta_h} + \sqrt{\alpha_h \beta_h}} = \frac{\epsilon}{2\beta_h} = \frac{\epsilon}{n}$$

- For any $\mathbf{z} \in \mathbb{R}^n$:

$$\begin{aligned}\text{prox}_{\frac{g}{\bar{L}}}(\mathbf{z}) =& \text{prox}_{\delta_{\mathbf{Ax}=\mathbf{b}}}(\mathbf{z}) \overset{\text{see th. 6.24 in [1]}}{=\!=} P_{\mathbf{Ax}=\mathbf{b}}(\mathbf{z}) \overset{\text{see lemma 6.26 in [1]}}{=\!=} \\ =& \mathbf{z} - \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}(\mathbf{A}\mathbf{z} - \mathbf{b})\end{aligned}$$

  where with $P_C(\mathbf{z})$ we denote the projection of $\mathbf{z}$ onto set C.

- The gradient of H_μ can be computed as below:

$$\begin{aligned}\nabla H_\mu(\mathbf{x}) =& \nabla h_\mu(\mathbf{x}) = \nabla M_h^\mu(\mathbf{x}) \overset{\text{see theor. 6.60 in [1]}}{=\!=} \\ =& \frac{1}{\mu}(\mathbf{x} - \text{prox}_{\mu h}(\mathbf{x})) \overset{\text{see example 6.8 in [1]}}{=\!=} \frac{1}{\mu}(\mathbf{x} - \mathcal{T}_\mu(\mathbf{x}))\end{aligned}$$

  where, for a description of $\mathcal{T}_\mu$, see section 6.*b*.

Consequently, the S-FISTA algorithm (adapted for problem (25)) turns into the following:

Algorithm 11: S-FISTA algorithm that solves (25).

Choose a random $\mathbf{x}_{rand} \in \mathbb{R}^n$, get its projection $\mathbf{x}_0 = \mathbf{P}_{\mathbf{Ax}=\mathbf{b}}(\mathbf{x}_{rand})$ and choose a value for stop criterion $\epsilon > 0$
$\mathbf{y}_0 \leftarrow \mathbf{x}_0$, $t_0 \leftarrow 1$, $\mu \leftarrow \frac{\epsilon}{n}$, $\bar{L} \leftarrow \frac{1}{\mu}$
$k \leftarrow 0$
while $h(\mathbf{x}) - h_{opt} > \epsilon$ **do**
  $\mathbf{x}_{k+1} \leftarrow P_{\mathbf{Ax}=\mathbf{b}}\left(\mathbf{y}_k - \frac{1}{\bar{L}}\left(\frac{1}{\mu}(\mathbf{x} - \mathcal{T}_\mu(\mathbf{x}))\right)\right)$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{x}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\mathbf{x}_{k+1} - \mathbf{x}_k)$
  $k \leftarrow k + 1$
end

Experimental results from the implementation, as well as the exact values that we set on the parameters will be explained in section 9.*d*.

Question 9.c. On this section we will attempt to solve problem (24) with the projected subgradient descent algorithm, as described in section 8.2.1 of [1]. We will test it with the Polyak stepsize, which is the optimal one.

The assumptions required by this algorithm are satisfied, since F is proper closed and convex function, and $\{\mathbf{Ax} = \mathbf{b}\}$ is a non-empty closed and convex set (if $n > m$). Moreover, in 9.*b* we explained that, for any $\mathbf{z} \in \mathbb{R}^n$:

$$\mathbf{P}_{\mathbf{Ax}=\mathbf{b}}(\mathbf{z}) = \mathbf{z} - \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}(\mathbf{A}\mathbf{z} - \mathbf{b})$$

Consequently, the projected subgradient method for the specific problem turns into:

Algorithm 12: Projected subgradient descent algorithm for the solution of (24).

Choose a random $\mathbf{x}_{rand} \in \mathbb{R}^n$ and get its projection $\mathbf{x}_0 = \mathbf{P}_{\mathbf{Ax}=\mathbf{b}}(\mathbf{x}_{rand})$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $h'(\mathbf{x}_k) \in \partial h(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{P}_{\mathbf{Ax}=\mathbf{b}}\left(\mathbf{x}_k - t_k h'(\mathbf{x}_k)\right)$
  $k \leftarrow k + 1$
end

- As a subgradient for $h(\mathbf{x})$ we will use the following weak result, which can be easily extracted from example 3.44 of [1]:

$$h'(\mathbf{x}) = \operatorname{sgn}(\mathbf{x})$$

- The Polyak step is:

$$t_k = \begin{cases} \frac{h(\mathbf{x}_k) - h_{opt}}{\|h'(\mathbf{x}_k)\|^2}, & h'(\mathbf{x}_k) \neq 0 \\ 1, & h'(\mathbf{x}_k) = 0 \end{cases}$$

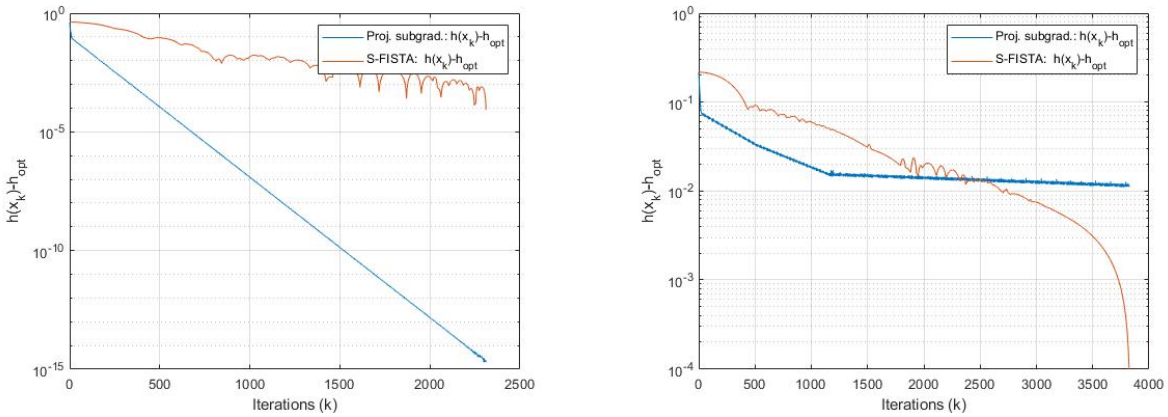
  where h_{opt} is the solution of the optimization problem (24), which was estimated by 'CVX'.

Experimental results from the implementation, as well as the exact values that we set on the parameters will be explained in section 9.*d*.

Question 9.d. On this section we implement both algorithms twice and we provide the corresponding graphs with their performance. The dimensions m, n of the problem are set as $m = 5, n = 10$. In order to be able to compare the algorithms it is necessary to set $n > m$, as in the opposite case, the system $\mathbf{Ax} = \mathbf{b}$ has one unique, or no solution. If there is no solution, then the problem (24) is impossible. Also, if there is one unique solution then the hyperplane $\mathbf{Ax} = \mathbf{b}$ consists of a single point, to which all algorithms converge after exactly 1 iteration (because every iteration requires a projection onto the hyperplane).

Moreover, the parameter of the loop-termination criterion for S-FISTA is set to $\epsilon = 10^{-4}$, while the projected subgradient algorithm is executed for as many iterations as were required for the S-FISTA. Matrices $\mathbf{A}$, $\mathbf{b}$, as well as the starting point $\mathbf{x}_0$ are drawn randomly.

Figures 26a, 26b demonstrate the quantity $h(\mathbf{x}_k) - h_{opt}$ versus the iterations k, for both algorithms. We observe that there are cases when the projected subgradient (with Polyak step) algorithm performs better, while in other cases the S-FISTA, which is an accelerated algorithm, converges faster. The optimization problem in both cases is the same, with the only change being the different values of $\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$.



(a) 1st execution. (b) 2nd execution.

Figure 26: Comparison between the projected subgradient and the S-FISTA algorithms.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex09.m`.

Exercise 10

Question 10.a, 10.e. As unit simplex is defined the subset of $\mathbb{R}^n$ such that:

$$\Delta_n = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{e}^T \mathbf{x} = 1, \mathbf{x} \geq \mathbf{0} \right\}$$

where $\mathbf{e}$ is the unit vector $[1, \ldots, 1]^T \in \mathbb{R}^n$.

The problem we have to solve in this exercise is:

$$\begin{array}{ll} \underset{\mathbf{x} \in \Delta_n}{\text{minimize}} & f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 \end{array} \tag{26}$$

As a first step we solve the problem via 'CVX' and we compute the optimal point $\mathbf{x}_*$ and the optimal value f_{opt}. We generate matrices $\mathbf{A}$, $\mathbf{b}$ twice. The first time their elements are drawn randomly, while the second time only the elements of $\mathbf{A}$ are drawn randomly, while $\mathbf{b}$ is generated in an exact way, as follows:

$$\mathbf{b} = \mathbf{A}\mathbf{x}_{true} + \mathbf{e}$$

where $\mathbf{e} \in \mathbb{R}^m$ is a white noise vector with a variance equal to 1.

Question 10.b. On this section we will utilize the projected subgradient descent algorithm, as described in section 8.2.1 of [1], in order to solve problem (26). Moreover, an adaptive stepsize will be employed (see theorem 8.30 in [1]), since the diameter of the unit simplex can be easily estimated.

The assumptions required by the algorithm are satisfied, since f is proper closed and convex function, and the unit simplex is a non-empty closed and convex set. Moreover, the projection onto the unit simplex, for any $\mathbf{x} \in \mathbb{R}^n$ is (see exercise 1 of this report):

$$\mathbf{P}_{\Delta_n}(\mathbf{x}) = [\mathbf{x} - \mu_* \mathbf{e}]_+, \text{ where } \mu_* \text{ is a solution of } \mathbf{e}^T [\mathbf{x} - \mu_* \mathbf{e}]_+ - 1 = 0. \tag{27}$$

Consequently, the projected subgradient method for the specific problem turns into:

Algorithm 13: Projected subgradient descent algorithm for the solution of (26).

$\mathbf{x}_0 \leftarrow \frac{1}{n}\mathbf{1} \in \Delta_n$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $f'(\mathbf{x}_k) \in \partial f(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{P}_{\Delta_n}(\mathbf{x}_k - t_k f'(\mathbf{x}_k))$
  $k \leftarrow k + 1$
end

- As a subgradient for $f(\mathbf{x})$ we will use the following weak result, which is proved by example 3.44 of [1]:

$$f'(\mathbf{x}) = \mathbf{A}^T \text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b})$$

- According to theorem 8.30 in [1], the adaptive stepsize can be chosen as:

$$t_k = \begin{cases} \frac{\sqrt{2\Theta}}{\|f'(\mathbf{x}_k)\|_2 \sqrt{k+1}}, & \text{if } f'(\mathbf{x}_k) \neq 0 \\ \frac{\sqrt{2\Theta}}{L_f \sqrt{k+1}}, & \text{if } f'(\mathbf{x}_k) = 0 \end{cases}$$

  where:

– Θ is an upper bound on the half-squared diameter of the subset of $\mathbb{R}^n$ from which $\mathbf{x}$ gets its values (i.e. the half-squared diameter of the unit simplex). That means:

$$\Theta \geq \max_{\mathbf{x},\mathbf{y}\in\Delta_n} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 = \max_{\mathbf{x},\mathbf{y}\in\Delta_n} \frac{1}{2}\left(\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 - \mathbf{x}^T\mathbf{y}\right) = \frac{1}{2}(1+1-0) = 1$$

– L_f is an upper bound to the ℓ_2-norms of the subgradients of f. As proved in section 4.*c* of this report, it is equal with:

$$L_f = \|\mathbf{A}\|_{2,2}\sqrt{m}$$

Experimental results from the implementation, as well as the exact values that we set on the parameters will be explained in section 10.*d*.

Question 10.c. On this section we will attempt to solve problem (26) with the mirror descent method, as described in section 9.1 of [1]. First of all, we observe that the requirements of the algorithm are satisfied, since f is a proper closed and convex function, and the unit simplex is a non-empty closed and convex set, where $\Delta_n \subseteq \mathbb{R}^n$.

Moreover, let's define an auxiliary function ω which satisfies some certain properties. Firstly, it has to be proper closed and convex. Also, it has to be differentiable over $\text{dom}(\partial\omega)$. Finally, the unit simplex must be a subset of her domain and $\omega + \delta_{\Delta_n}$ has to be a σ-strongly convex function ($\sigma > 0$). Function δ_C has already been defined in section 9.*b* of this report.

The general form of the algorithm (applied onto the unit simplex) can be seen below:

Algorithm 14: Mirror descent algorithm.

Choose a random $\mathbf{x}_0 \in \Delta_n \cap \text{dom}(\partial\omega)$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $f'(\mathbf{x}_k) \in \partial f(\mathbf{x}_k)$
  $\mathbf{x}_{k+1} \leftarrow \text{argmin}_{\mathbf{x}\in\Delta_n} \{\langle t_k f'(\mathbf{x}_k) - \nabla\omega(\mathbf{x}_k), \mathbf{x}\rangle + \omega(\mathbf{x})\}$
  $k \leftarrow k+1$
end

- There are various options for selecting function ω. For example, if we set $\omega(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2$ then the algorithm turns into the projected subgradient one. On this section, we will set:

$$\omega(\mathbf{x}) = \begin{cases} \sum_{i=1}^m x_i \log x_i, & \text{if } x_i \in \mathbb{R}_+^n \\ \infty, & \text{if } x_i \notin \mathbb{R}_+^n \end{cases}$$

  Note: Assume that $0\log 0 = 0$. Then ω is a $(\sigma = 1)$-strongly convex function (with respect to the ℓ_1-norm), according to example 5.27 in [1], and all requirements for that function are satisfied.

  As explained by example 9.10 of [1], for the specific ω the update equation of the mirror descent algorithm turns into:

$$x_{(k+1),i} \leftarrow \frac{x_{k,i}e^{-t_k f_i'(\mathbf{x}_k)}}{\sum_{j=1}^n x_{k,j}e^{-t_k f_j'(\mathbf{x}_k)}}, \quad i = 1,\ldots,n$$

  where $f_i'(\mathbf{x}_k)$ is the i-th element of $f'(\mathbf{x}_k)$

- As a subgradient for $f(\mathbf{x})$ we will use the following weak result, which is proved by example 3.44 of [1]:

$$f'(\mathbf{x}) = \mathbf{A}^T \text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b})$$

- Again, we will use an adaptive stepsize (as defined by theorem 9.18 in [1]). The dual of the ℓ_1-norm is the infinity norm, so we get:

$$t_k = \begin{cases} \frac{\sqrt{2\sigma}}{\|f'(\mathbf{x}_k)\|_\infty \sqrt{k+1}}, & \text{if } f'(\mathbf{x}_k) \neq 0 \\ \frac{\sqrt{2\sigma}}{L_f \sqrt{k+1}}, & \text{if } f'(\mathbf{x}_k) = 0 \end{cases}$$

  where $L_f \geq \|f'(\mathbf{x})\|_\infty, \ \forall \mathbf{x} \in \mathbb{R}^n$. However, we have already set $\|f'(\mathbf{x})\|_\infty = \|\mathbf{A}^T \text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_\infty \leq \||\mathbf{A}^T|\mathbf{e}\|_\infty$, so we can set $L_f = \||\mathbf{A}^T|\mathbf{e}\|_\infty$.

Consequently, the mirror descent method for the specific problem turns into:

Algorithm 15: Mirror descent algorithm for the solution of (26).

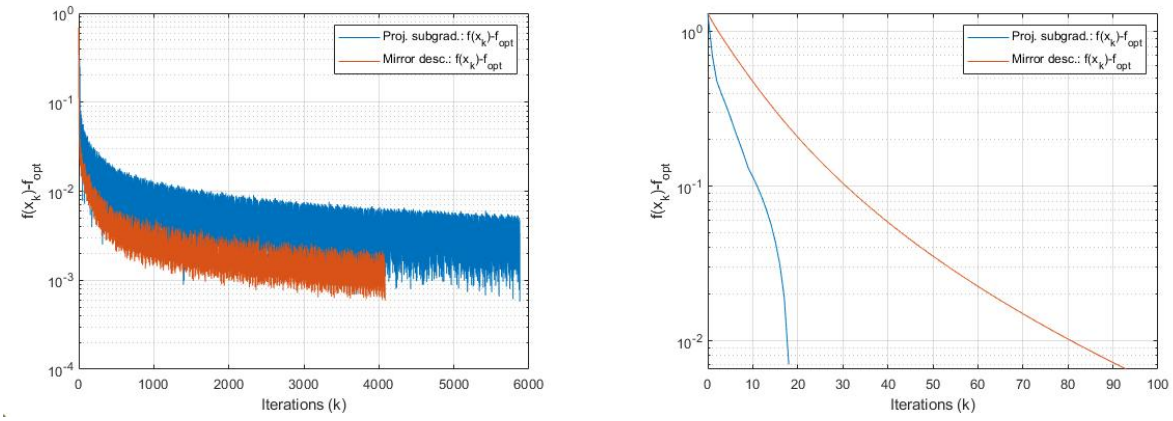
$\mathbf{x}_0 \leftarrow \frac{1}{n}\mathbf{1} \in \Delta_n$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  Choose a stepsize $t_k > 0$ and a subgradient $f'(\mathbf{x}_k) \in \partial f(\mathbf{x}_k)$
  $x_{(k+1),i} \leftarrow \frac{x_{k,i} e^{-t_k f_i'(\mathbf{x}_k)}}{\sum_{j=1}^n x_{k,j} e^{-t_k f_j'(\mathbf{x}_k)}}, \quad i = 1, \ldots, n$
  $k \leftarrow k + 1$
end

Experimental results from the implementation, as well as the exact values that we set on the parameters, will be explained in section 10.*d*.

Question 10.d, 10.e. On this section we implement both algorithms and we provide the corresponding graphs with their performance. The dimensions m, n of the problem are set as $m = 5, n = 10$. Moreover, the parameter c of the loop-termination criterion is set to $c = 1.001$.

We study 2 different cases of matrices $\mathbf{A}$, $\mathbf{b}$. At figure 27a they are drawn randomly from the Uniform distribution, while at figure 27b they are constructed in an exact way as follows: Firstly, matrix $\mathbf{A}$ is drawn from the Uniform distribution (as before), and then $\mathbf{b}$ is constructed to be equal with $\mathbf{b} = \mathbf{A}\mathbf{x}_{true} + \mathbf{e}$, where $\mathbf{x}_{true}$ was drawn randomly and $\mathbf{e}$ is a white noise vector, with variance equal to 1.

Figures 27a, 27b demonstrate the quantity $f(\mathbf{x}_k) - f_{opt}$ versus the iterations k, for both algorithms. When $\mathbf{A}$, $\mathbf{b}$ are drawn randomly, then the mirror descent performs better. However, when $\mathbf{A}$, $\mathbf{b}$ are drawn in an exact way, then the projected subgradient algorithm is the faster. Moreover, we observe that, when $\mathbf{A}$, $\mathbf{b}$ are drawn in an exact way, fewer iterations are required for convergence, compared to randomly drawn $\mathbf{A}$, $\mathbf{b}$.



(a) Random $\mathbf{A}, \mathbf{b}$. (b) Exact $\mathbf{A}, \mathbf{b}$, s.t. $\mathbf{A}\mathbf{x}_{true} = \mathbf{b}$.

Figure 27: Projected subgradient descent & mirror descent for $m = 5$, $n = 10$.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex10.m`.

Exercise 11

The problem we have to solve in this exercise is:

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{28}$$

As a first step we solve the problem via 'CVX' and we compute the optimal point $\mathbf{x}_*$ and the optimal value f_{opt}. We generate matrices $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ by drawing them from the Uniform distribution and we set $m = 5$, $n = 50$.

Question 11.a. Initially, we have to transform problem (28) into an equivalent one:

$$\begin{aligned} \underset{\mathbf{x}\in\text{dom}f,\,\mathbf{z}\in\mathbb{R}^n}{\text{minimize}} \quad & F(\mathbf{x},\mathbf{z}) = f(\mathbf{x}) + g(\mathbf{z}) = \mathbf{c}^T\mathbf{x} + \delta_{\mathbb{R}^n_+}(\mathbf{z}), \;\; \text{with dom}f = \{\mathbf{x}\in\mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}\} \\ \text{subject to} \quad & \mathbf{x} = \mathbf{z} \end{aligned} \tag{29}$$

where function δ_C has already been defined in section 9.*b* of this report. Both function f and g are proper closed and convex.

The augmented Lagrangian function L_ρ ($\rho > 0$) for problem (29) is (see remark 15.1 in [1]):

$$\begin{aligned} L_\rho(\mathbf{x},\mathbf{z},\mathbf{y}) =& f(\mathbf{x}) + g(\mathbf{z}) + \langle \mathbf{y}, \mathbf{x}-\mathbf{z}\rangle + \frac{\rho}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{x}-\mathbf{z}) + \frac{\rho}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 = \\ =& \mathbf{c}^T\mathbf{x} + \delta_{\mathbb{R}^n_+}(\mathbf{z}) + \frac{\rho}{2}\left\|\mathbf{x}-\mathbf{z}+\frac{1}{\rho}\mathbf{y}\right\|_2^2 - \frac{1}{2}\|\mathbf{y}\|_2^2 \end{aligned}$$

where $\mathbf{y}\in\mathbb{R}^n$.

The ADMM updates can be written as follows (see section 15.2 of [1]):

$$\begin{aligned} \mathbf{x}_{k+1} \in& \text{argmin}_{\mathbf{x}\in\text{dom}f} L_\rho(\mathbf{x},\mathbf{z}_k,\mathbf{y}_k) = \text{argmin}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{c}^T\mathbf{x} + \delta_{\mathbb{R}^n_+}(\mathbf{z}_k) + \frac{\rho}{2}\left\|\mathbf{x}-\mathbf{z}_k+\frac{1}{\rho}\mathbf{y}_k\right\|_2^2 - \frac{1}{2}\|\mathbf{y}_k\|_2^2\right\} \\ =& \text{argmin}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{c}^T\mathbf{x} + \frac{\rho}{2}\left\|\mathbf{x}-\mathbf{z}_k+\frac{1}{\rho}\mathbf{y}_k\right\|_2^2\right\} \end{aligned} \tag{30}$$

$$\begin{aligned} \mathbf{z}_{k+1} \in& \text{argmin}_{\mathbf{z}} L_\rho(\mathbf{x}_{k+1},\mathbf{z},\mathbf{y}_k) = \text{argmin}_{\mathbf{z}}\left\{\mathbf{c}^T\mathbf{x}_{k+1} + \delta_{\mathbb{R}^n_+}(\mathbf{z}) + \frac{\rho}{2}\left\|\mathbf{x}_{k+1}-\mathbf{z}+\frac{1}{\rho}\mathbf{y}_k\right\|_2^2 - \frac{1}{2}\|\mathbf{y}_k\|_2^2\right\} = \\ =& \text{argmin}_{\mathbf{z}}\left\{\delta_{\mathbb{R}^n_+}(\mathbf{z}) + \frac{\rho}{2}\left\|\mathbf{x}_{k+1}-\mathbf{z}+\frac{1}{\rho}\mathbf{y}_k\right\|_2^2\right\} \end{aligned} \tag{31}$$

$$\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k + \rho\left(\mathbf{x}_{k+1} - \mathbf{z}_{k+1}\right) \tag{32}$$

Question 11.b. As a first step before the implementation, we have to solve the optimization subproblems given by equations (30) and (31).

- The solution of problem (30) is given by:

$$\mathbf{x}_{k+1} \in \text{argmin}_{\mathbf{x}\in \text{dom} f}\left\{\mathbf{c}^T\mathbf{x} + \frac{\rho}{2}\left\|\mathbf{x} - \mathbf{z}_k + \frac{1}{\rho}\mathbf{y}_k\right\|_2^2\right\} = -\frac{1}{\rho}\mathbf{c} + \mathbf{z}_k - \frac{1}{\rho}\mathbf{y}_k =$$
$$= \left(\mathbf{I}_n - \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\right)\left(\mathbf{z}_k - \frac{1}{\rho}(\mathbf{c} + \mathbf{y}_k)\right) + \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{b}$$

Proof: The problem can be expressed as:

$$\begin{array}{ll}\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f_0(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{\rho}{2}\left\|\mathbf{x} - \left(\mathbf{z}_k - \frac{1}{\rho}\mathbf{y}_k\right)\right\|_2^2 \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b}\end{array}$$

Let's denote $\mathbf{q} = \mathbf{z}_k - \frac{1}{\rho}\mathbf{y}_k$. Then the Slater condition is satisfied (obviously) and, if we assume $\mathbf{v} \in \mathbb{R}^m$, the KKT conditions are:

$$\nabla f_0(\mathbf{x}_*) + \mathbf{A}^T\mathbf{v} = \mathbf{0} \Leftrightarrow \mathbf{c} + \rho(\mathbf{x}_* - \mathbf{q}) + \mathbf{A}^T\mathbf{v} = \mathbf{0} \Leftrightarrow \mathbf{x}_* = \mathbf{q} - \frac{1}{\rho}\left(\mathbf{c} + \mathbf{A}^T\mathbf{v}\right) \tag{33}$$

$$\mathbf{A}\mathbf{x}_* = \mathbf{b} \overset{(33)}{\Leftrightarrow} \mathbf{A}\left(\mathbf{q} - \frac{1}{\rho}\left(\mathbf{c} + \mathbf{A}^T\mathbf{v}\right)\right) = \mathbf{b} \Leftrightarrow \mathbf{v} = \left(\mathbf{A}\mathbf{A}^T\right)^{-1}\left(\mathbf{A}\left(\rho\mathbf{q} - \mathbf{c}\right) - \rho\mathbf{b}\right) \tag{34}$$

Then, if we substitute (34) into (33) we get:

$$\begin{aligned}\mathbf{x}_* =& \mathbf{q} - \frac{1}{\rho}\left(\mathbf{c} + \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\left(\mathbf{A}\rho\mathbf{q} - \mathbf{A}\mathbf{c} - \rho\mathbf{b}\right)\right) = \\ =& \mathbf{z}_k - \frac{1}{\rho}\mathbf{y}_k - \frac{1}{\rho}\left(\mathbf{c} + \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\left(\mathbf{A}\rho\left(\mathbf{z}_k - \frac{1}{\rho}\mathbf{y}_k\right) - \mathbf{A}\mathbf{c} - \rho\mathbf{b}\right)\right) = \\ =& \left(\mathbf{I}_n - \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\right)\left(\mathbf{z}_k - \frac{1}{\rho}(\mathbf{c} + \mathbf{y}_k)\right) + \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{b}\end{aligned}$$

- The solution of problem (31) is given by:

$$\mathbf{z}_{k+1} \in \text{argmin}_{\mathbf{z}}\left\{\delta_{\mathbb{R}^n_+}(\mathbf{z}) + \frac{\rho}{2}\left\|\mathbf{x}_{k+1} - \mathbf{z} + \frac{1}{\rho}\mathbf{y}_k\right\|_2^2\right\} =$$
$$= \text{argmin}_{\mathbf{z}}\left\{\delta_{\mathbb{R}^n_+}(\mathbf{z}) + \frac{\rho}{2}\left\|\mathbf{z} - \left(\mathbf{x}_{k+1} + \frac{1}{\rho}\mathbf{y}_k\right)\right\|_2^2\right\} = \left[\mathbf{x}_{k+1} + \frac{1}{\rho}\mathbf{y}_k\right]_+$$

where $[\mathbf{x}]_+ = \left[x_i^+\right]_{i=1}^n$, s.t. $x_i^+ = \begin{cases}0, & \text{if } x_i^+ \le 0 \\ x_i^+ & \text{if } x_i^+ > 0\end{cases}$.

Proof: Let's denote $\mathbf{q} = \mathbf{x}_{k+1} + \frac{1}{\rho}\mathbf{y}_k$. Then, the problem can be expressed as:

$$\begin{array}{ll}\underset{\mathbf{z}\in\mathbb{R}^n}{\text{minimize}} & f_0(\mathbf{z}) = \frac{\rho}{2}\|\mathbf{z} - \mathbf{q}\|_2^2 \\ \text{subject to} & f_i(\mathbf{z}) = -z_i \le 0, \quad i = 1, \ldots, n\end{array}$$

The Slater condition is satisfied (obviously), so if we assume $\mathbf{u} \in \mathbb{R}^n$ the KKT conditions are:

$$\nabla f_0(\mathbf{z}_*) + \sum_{i=1}^{n} u_i \nabla f_i(\mathbf{z}_*) = 0 \Leftrightarrow z_i^* = \frac{u_i}{\rho} + b_i, \quad i = 1, \ldots, n$$
$$\mathbf{z}_* \geq \mathbf{0}$$
$$\mathbf{u} \geq \mathbf{0}$$
$$u_i f_i(\mathbf{z}_*) = 0 \Leftrightarrow u_i z_i^* = 0, \quad i = 1, \ldots, n$$

The solution of the KKT conditions (the proof is trivial and it is ommited) is:

$$\mathbf{z}_* = [\mathbf{q}]_+ = \left[\mathbf{x}_{k+1} + \frac{1}{\rho}\mathbf{y}_k\right]_+$$

Questions 11.c, 11.d. The ADMM algorithm which solves optimization problem (29) is the following (adapted from its generic form which is presented in section 15.2 of [1]):

Algorithm 16: ADMM algorithm for the solution of (29).

Choose a random $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{z}_0 \in \mathbb{R}^n$, $\mathbf{y}_0 \in \mathbb{R}^n$ $k \leftarrow 0$
while *the iterations haven't converged* **do**
  $x_{k+1} \leftarrow \text{argmin}_{\mathbf{x} \in \text{dom} f} \left\{ \mathbf{c}^T\mathbf{x} + \frac{\rho}{2} \left\| \mathbf{x} - \mathbf{z}_k + \frac{1}{\rho}\mathbf{y}_k \right\|_2^2 \right\}$
  $z_{k+1} \leftarrow \text{argmin}_{\mathbf{z}} \left\{ \delta_{\mathbb{R}_+^n}(\mathbf{z}) + \frac{\rho}{2} \left\| \mathbf{x}_{k+1} - \mathbf{z} + \frac{1}{\rho}\mathbf{y}_k \right\|_2^2 \right\}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k + \rho\left(\mathbf{x}_{k+1} - \mathbf{z}_{k+1}\right)$
  $k \leftarrow k + 1$
end

The optimization sub-problems for the updates of $\mathbf{x}$, $\mathbf{z}$ have already been solved in section 11.*b* of this report. Moreover, a very efficient implementation is possible for the update of $\mathbf{x}$. More specifically, product $\mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\mathbf{s}$, where $\mathbf{s} \in \mathbb{R}^n$ requires the computation of high-dimensional *nxn* matrices. However, we can separate it as follows:

$$\mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\mathbf{s} = \left\{\mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\right\}\{\mathbf{A}\mathbf{s}\}$$

Then, we have to compute the product of 2 *mxm* matrices. Such a product is much more efficient to compute, since $m << n$.

As a termination criterion for the iterations we assume the following one:

$$\left|F(\mathbf{x}, \mathbf{z}) - F_{opt}\right| \leq \epsilon, \text{ with } \epsilon = 10^{-4}.$$

Figure 28 shows the performance of ADMM while it solves problem (29). Matrices $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$ were drawn from the Uniform distribution with dimensions $m = 5$, $n = 500$. Also, parameter ρ was set to be equal with 20.

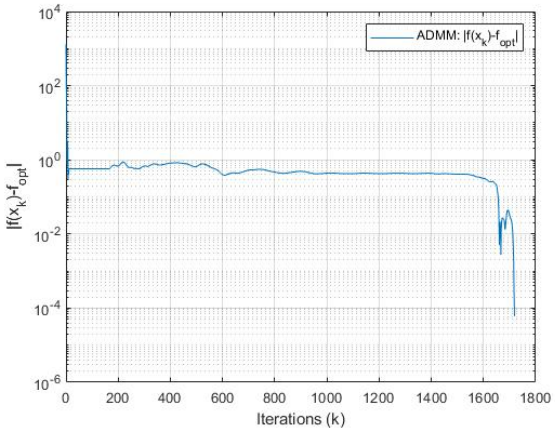


Figure 28: ADMM for the solution of (29).

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex11.m`.

Exercise 12

Question 12.a. On this section we have to compute the projection of a random vector $\mathbf{d} \in \mathbb{R}^n$ onto $\mathcal{S}_1 = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ (ensured that it is a nonempty set).

The projection is the solution of the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

If we set $g(\mathbf{w}) = \delta_{\{\mathbf{z}:\mathbf{z} \leq \mathbf{b}\}}(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \leq \mathbf{b} \\ \infty, & \text{elsewhere} \end{cases}$, for any $\mathbf{w} \in \mathbb{R}^m$, then the above problem is equivalent with:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{A}\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 + \delta_{\{\mathbf{z}:\mathbf{z} \leq \mathbf{b}\}}(\mathbf{A}\mathbf{x}) \tag{35}$$

We will implement the Dual Proximal Gradient (DPG) method in order to solve the above problem. More in detail, all requirements of this algorithm are satisfied, since f is proper closed and σ-strongly convex with $\sigma = 1$ (see example 5.19 in [1]). Also, g is proper closed and convex and $\mathbf{A}\mathbf{x}$ is a linear transformation of $\mathbf{x}$. Finally, as stated above the set $\mathcal{S}_1$ is nonempty.

If we denote with $\mathcal{A}(\mathbf{x})$ a linear transformation of $\mathbf{x}$, we get the general form of the DPG algorithm which is:

Algorithm 17: DPG algorithm - primal representation.

Choose a random $\mathbf{y}_0 \in \text{dom} g$ and a parameter $L \geq \frac{\|\mathcal{A}\|_2^2}{\sigma}$
$k \leftarrow 0$
while *the iterations haven't converged* **do**
  $\mathbf{x}_k \leftarrow \text{argmax}_{\mathbf{x} \in \text{dom} f} \left\{\langle \mathbf{x}, \mathcal{A}^T(\mathbf{y}_k)\rangle - f(\mathbf{x})\right\}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k - \frac{1}{L}\mathcal{A}(\mathbf{x}_k) + \frac{1}{L}\text{prox}_{Lg}\left(\mathcal{A}(\mathbf{x}_k) - L\mathbf{y}_k\right)$
  $k \leftarrow k + 1$
end

On our problem, the algorithm is adapted as follows:

- $\mathcal{A}(\mathbf{x}_k) = \mathbf{A}\mathbf{x}_k$, $\mathcal{A}^T(\mathbf{y}_k) = \mathbf{A}^T\mathbf{y}_k$, $\|\mathcal{A}\|_2^2 = \|\mathbf{A}\|_2^2$
- $\text{argmax}_{\mathbf{x} \in \text{dom} f} \left\{\langle \mathbf{x}, \mathcal{A}^T(\mathbf{y}_k)\rangle - f(\mathbf{x})\right\} = \text{argmax}_{\mathbf{x} \in \text{dom} f} \left\{\mathbf{x}^T\mathbf{A}^T\mathbf{y}_k - \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2\right\} =$
  $= \text{argmax}_{\mathbf{x} \in \text{dom} f} \left\{\mathbf{x}^T\mathbf{A}^T\mathbf{y}_k - \frac{1}{2}\mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{d} - \frac{1}{2}\mathbf{d}^T\mathbf{d}\right\} =$
  $= \text{argmax}_{\mathbf{x} \in \text{dom} f} \left\{-\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d} + \mathbf{A}^T\mathbf{y}_k\right)\right\|_2^2\right\} = \text{argmin}_{\mathbf{x} \in \text{dom} f} \left\{\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d} + \mathbf{A}^T\mathbf{y}_k\right)\right\|_2^2\right\} =$
  $= \mathbf{d} + \mathbf{A}^T\mathbf{y}_k$
- $\text{prox}_{Lg}(\mathbf{z}) = \text{prox}_{L\delta_{\{\mathbf{z}:\mathbf{z} \leq \mathbf{b}\}}}(\mathbf{z}) = \text{prox}_{\delta_{\{\mathbf{z}:\mathbf{z} \leq \mathbf{b}\}}}(\mathbf{z}) \overset{\text{see th. 6.24 in [1]}}{=\!=} \text{Proj}_{\{\mathbf{z}:\mathbf{z} \leq \mathbf{b}\}}(\mathbf{z}) =$
  $\overset{\text{see lemma 6.26 in [1]}}{=\!=\!=} [\min(z_i, b_i)]_{i=1}^n$
- As a termination criterion for the iterations (in order to detect convergence) we can use the following (with f_{opt} estimated by 'CVX' software):

$$f(\mathbf{x}) - f_{opt} \leq \epsilon$$

Finally, the DPG algorithm that solves problem (35) turns into:

Algorithm 18: DPG (primal representation) that solves problem (35).

Choose a random $\mathbf{y}_0 \in \mathbb{R}^m$
$L \leftarrow \frac{\|\mathbf{A}\|_2^2}{\sigma}$, $k \leftarrow 0$
while $(k = 0)$ *or* $(f(\mathbf{x}_k) - f_{opt} > \epsilon)$ **do**
  $\mathbf{x}_k \leftarrow \mathbf{d} + \mathbf{A}^T\mathbf{y}_k$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k - \frac{1}{L}\mathbf{A}\mathbf{x}_k + \frac{1}{L}\left[\min\left((\mathbf{A}\mathbf{x}_k)_{(i)} - L\mathbf{y}_{k(i)}, b_i\right)\right]_{i=1}^{m}$
  $k \leftarrow k + 1$
end

Figure 29 plots the quantity $f(\mathbf{x}_k) - f_{opt}$, while the algorithm converges to the solution of problem (35) (which is the projection of $\mathbf{d}$ onto $\mathcal{S}_1$). f_{opt} was obtained by solving the same problem with 'CVX' software. Furthermore, matrices $\mathbf{A}, \mathbf{d}$ were drawn randomly, while $\mathbf{b}$ was constructed to be equal with $\mathbf{A}\mathbf{x}_{true}$, with $\mathbf{x}_{true} \in \mathbb{R}^n$ randomly drawn too, in order to be also applicable at the next parts of this exercise. Finally, the parameter ϵ was set to be equal with 10^{-9}.

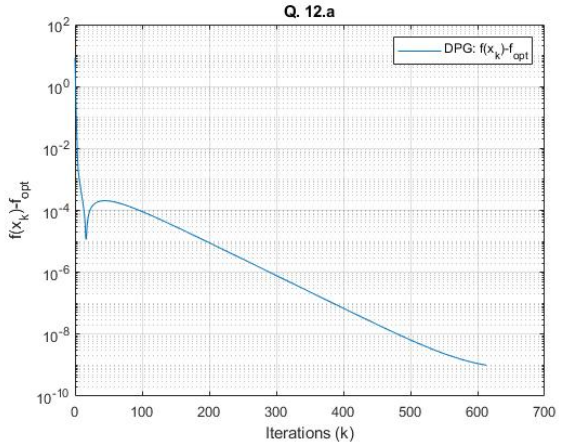


Figure 29: DPG for obtaining the projection of a vector onto $\mathcal{S}_1$.

Question 12.b. On this section we will compute the projection of a random vector $\mathbf{d} \in \mathbb{R}^n$ onto $\mathcal{S}_2 = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{v}_1 \leq \mathbf{x} \leq \mathbf{v}_2\}$ (ensured that it is a nonempty set).

The projection is the solution of the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{v}_1 \leq \mathbf{x}, \\ & \mathbf{x} \leq \mathbf{v}_2 \end{aligned}$$

Before we continue, let's denote with $\mathcal{S}_{2.1} = \{\mathbf{z} \in \mathbb{R}^m | \mathbf{z} = \mathbf{b}\}$, $\mathcal{S}_{2.2} = \{\mathbf{z} \in \mathbb{R}^n | \mathbf{v}_1 \leq \mathbf{z}\}$ and $\mathcal{S}_{2.3} = \{\mathbf{z} \in \mathbb{R}^n | \mathbf{z} \leq \mathbf{v}_2\}$.

Then, assume $\mathbf{z}_1 \in \mathbb{R}^m$, $\mathbf{z}_2, \mathbf{z}_3 \in \mathbb{R}^n$. We set $g\left(\begin{bmatrix}\mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3\end{bmatrix}\right) = \delta_{\mathcal{S}_{2.1}}(\mathbf{z}_1) + \delta_{\mathcal{S}_{2.2}}(\mathbf{z}_2) +$

$\delta_{\mathcal{S}_{2.3}}(\mathbf{z}_3)$, where $\delta_{\mathcal{S}_{2.i}}(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \mathcal{S}_{2.i} \\ \infty, & \text{elsewhere} \end{cases}$.

Consequently, we can turn the above optimization problem into the following equivalent one:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{A}_0\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 + \delta_{\mathcal{S}_{2.1}}(\mathbf{A}\mathbf{x}) + \delta_{\mathcal{S}_{2.2}}(\mathbf{x}) + \delta_{\mathcal{S}_{2.3}}(\mathbf{x}) \tag{36}$$

where $\mathbf{A}_0 = \begin{bmatrix} \mathbf{A} \\ \mathbf{I}_n \\ \mathbf{I}_n \end{bmatrix}$.

Again, the Dual Proximal Gradient (DPG) method will be used in order to solve the above problem. More in detail, all requirements of the algorithm are satisfied, since f is proper closed and σ-strongly convex with $\sigma = 1$ (see example 5.19 in [1]). Also, g is proper closed and convex and $\mathbf{A}_0\mathbf{x}$ is a linear transformation of $\mathbf{x}$. Finally, as stated above the set $\mathcal{S}_2$ is nonempty.

The general form of the DPG algorithm can be found in section 12.*a*. For our specific problem, the algorithm is adapted as follows:

- $\mathcal{A}(\mathbf{x}_k) = \mathbf{A}_0\mathbf{x}_k$, $\mathcal{A}^T(\mathbf{y}_k) = \mathbf{A}_0^T\mathbf{y}_k$, $\|\mathcal{A}\|_2^2 = \|\mathbf{A}_0\|_2^2$
- $\text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\langle \mathbf{x}, \mathcal{A}^T(\mathbf{y}_k)\rangle - f(\mathbf{x})\right\} = \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{x}^T\mathbf{A}_0^T\mathbf{y}_k - \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2\right\} =$
  $= \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{x}^T\mathbf{A}_0^T\mathbf{y}_k - \frac{1}{2}\mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{d} - \frac{1}{2}\mathbf{d}^T\mathbf{d}\right\} =$
  $= \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{-\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d} + \mathbf{A}_0^T\mathbf{y}_k\right)\right\|_2^2\right\} = \text{argmin}_{\mathbf{x}\in\text{dom}f}\left\{\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d} + \mathbf{A}_0^T\mathbf{y}_k\right)\right\|_2^2\right\} =$
  $= \mathbf{d} + \mathbf{A}_0^T\mathbf{y}_k$
- $\text{prox}_{Lg}(\mathbf{z}) = \text{prox}_{L\sum\delta}(\mathbf{z}) = \text{prox}_{\sum\delta}(\mathbf{z}) \overset{\text{see rem. 6.7 in [1]}}{=\!=} \begin{bmatrix} \text{prox}_{\delta_{\mathcal{S}_{2.1}}}(\mathbf{z}_{1:m}) \\ \text{prox}_{\delta_{\mathcal{S}_{2.2}}}(\mathbf{z}_{m+1:m+n}) \\ \text{prox}_{\delta_{\mathcal{S}_{2.3}}}(\mathbf{z}_{m+n+1:m+2n}) \end{bmatrix} =$
  $\overset{\text{see th. 6.23 in [1]}}{=\!=} \begin{bmatrix} \text{Proj}_{\mathcal{S}_{2.1}}(\mathbf{z}_{1:m}) \\ \text{Proj}_{\mathcal{S}_{2.2}}(\mathbf{z}_{m+1:m+n}) \\ \text{Proj}_{\mathcal{S}_{2.3}}(\mathbf{z}_{m+n+1:m+2n}) \end{bmatrix} \overset{\text{see lemma 6.26 in [1]}}{=\!=} \begin{bmatrix} \mathbf{b} \\ \left[\max(z_{m+i}, v_{1(i)})\right]_{i=1}^{n} \\ \left[\min(z_{m+n+i}, v_{2(i)})\right]_{i=1}^{n} \end{bmatrix}$
- As a termination criterion for the iterations (in order to detect convergence) we can use the following (with f_{opt} estimated by 'CVX' software):

$$f(\mathbf{x}) - f_{opt} \leq \epsilon$$

Finally, the DPG algorithm that solves problem (36) turns into:

Algorithm 19: DPG (primal representation) that solves problem (36).

Choose a random $\mathbf{y}_0 \in \mathbb{R}^m$

$L \leftarrow \frac{\|\mathbf{A}_0\|_2^2}{\sigma}$, $k \leftarrow 0$

while $(k = 0)$ *or* $(f(\mathbf{x}_k) - f_{opt} > \epsilon)$ **do**

$\mathbf{x}_k \leftarrow \mathbf{d} + \mathbf{A}_0^T \mathbf{y}_k$

$$\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k - \frac{1}{L}\mathbf{A}_0\mathbf{x}_k + \frac{1}{L}\begin{bmatrix} \mathbf{b} \\ \left[\max\left(\mathbf{x}_{k(i)} - L\mathbf{y}_{k(m+i)}, v_{1(i)}\right)\right]_{i=1}^{n} \\ \left[\min\left(\mathbf{x}_{k(i)} - L\mathbf{y}_{k(m+n+i)}, v_{2(i)}\right)\right]_{i=1}^{n} \end{bmatrix}$$

$k \leftarrow k + 1$

end

Figure 30 plots the quantity $f(\mathbf{x}_k) - f_{opt}$, while the algorithm converges to the solution of problem (36) (which is the projection of $\mathbf{d}$ onto $\mathcal{S}_2$). f_{opt} was obtained by solving the same problem with 'CVX' software. Furthermore, matrices $\mathbf{A}, \mathbf{b}, \mathbf{d}$ are the same with the respective matrices from section 12.*a*. Also, $\mathbf{v}_1, \mathbf{v}_2$ were drawn randomly. Finally, the parameter ϵ was set to be equal with 10^{-9}.

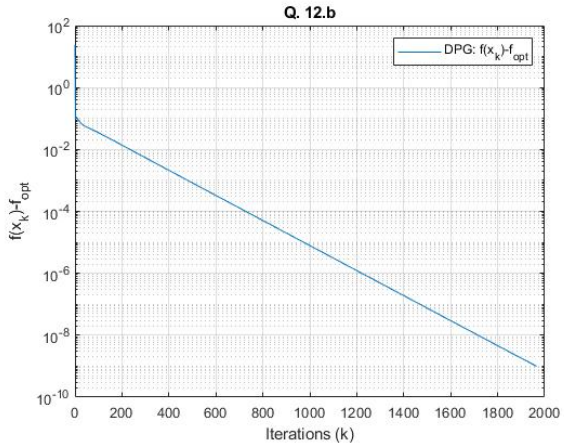


Figure 30: DPG for obtaining the projection of a vector onto $\mathcal{S}_2$.

Question 12.c. On this section we will compute the projection of a random vector $\mathbf{d} \in \mathbb{R}^n$ onto $\mathcal{S}_3 = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{v}_1 \leq \mathbf{x} \leq \mathbf{v}_2\}$ (ensured that it is a nonempty set).

The projection is the solution of the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{v}_1 \leq \mathbf{x}, \\ & \mathbf{x} \leq \mathbf{v}_2 \end{aligned}$$

Before we continue, let's denote with $\mathcal{S}_{3.1} = \{\mathbf{z} \in \mathbb{R}^m | \mathbf{z} \leq \mathbf{b}\}$, $\mathcal{S}_{3.2} = \{\mathbf{z} \in \mathbb{R}^n | \mathbf{v}_1 \leq \mathbf{z}\}$ and $\mathcal{S}_{3.3} = \{\mathbf{z} \in \mathbb{R}^n | \mathbf{z} \leq \mathbf{v}_2\}$.

Then, assume $\mathbf{z}_1 \in \mathbb{R}^m$, $\mathbf{z}_2, \mathbf{z}_3 \in \mathbb{R}^n$. We set $g\left(\begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix}\right) = \delta_{\mathcal{S}_{3.1}}(\mathbf{z}_1) + \delta_{\mathcal{S}_{3.2}}(\mathbf{z}_2) +$

$\delta_{\mathcal{S}_{3.3}}(\mathbf{z}_3)$, where $\delta_{\mathcal{S}_{3.i}}(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \mathcal{S}_{3.i} \\ \infty, & \text{elsewhere} \end{cases}$.

Consequently, we can turn the above optimization problem into the following equivalent one:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{A}_0\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 + \delta_{\mathcal{S}_{3.1}}(\mathbf{A}\mathbf{x}) + \delta_{\mathcal{S}_{3.2}}(\mathbf{x}) + \delta_{\mathcal{S}_{3.3}}(\mathbf{x}) \tag{37}$$

where $\mathbf{A}_0 = \begin{bmatrix} \mathbf{A} \\ \mathbf{I}_n \\ \mathbf{I}_n \end{bmatrix}$.

Again, the Dual Proximal Gradient (DPG) method will be used in order to solve the above problem. More in detail, all requirements of the algorithm are satisfied, since f is proper closed and σ-strongly convex with $\sigma = 1$ (see example 5.19 in [1]). Also, g is proper closed and convex and $\mathbf{A}_0\mathbf{x}$ is a linear transformation of $\mathbf{x}$. Finally, as stated above the set $\mathcal{S}_3$ is nonempty.

The general form of the DPG algorithm can be found in section 12.*a*. For our specific problem, the algorithm is adapted as follows:

- $\mathcal{A}(\mathbf{x}_k) = \mathbf{A}_0\mathbf{x}_k$, $\mathcal{A}^T(\mathbf{y}_k) = \mathbf{A}_0^T\mathbf{y}_k$, $\|\mathcal{A}\|_2^2 = \|\mathbf{A}_0\|_2^2$
- $\text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\langle \mathbf{x}, \mathcal{A}^T(\mathbf{y}_k)\rangle - f(\mathbf{x})\right\} = \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{x}^T\mathbf{A}_0^T\mathbf{y}_k - \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2\right\} =$
  $= \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{x}^T\mathbf{A}_0^T\mathbf{y}_k - \frac{1}{2}\mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{d} - \frac{1}{2}\mathbf{d}^T\mathbf{d}\right\} =$
  $= \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{-\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d} + \mathbf{A}_0^T\mathbf{y}_k\right)\right\|_2^2\right\} = \text{argmin}_{\mathbf{x}\in\text{dom}f}\left\{\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d} + \mathbf{A}_0^T\mathbf{y}_k\right)\right\|_2^2\right\} =$
  $= \mathbf{d} + \mathbf{A}_0^T\mathbf{y}_k$
- $\text{prox}_{Lg}(\mathbf{z}) = \text{prox}_{L\sum\delta}(\mathbf{z}) = \text{prox}_{\sum\delta}(\mathbf{z}) \overset{\text{see rem. 6.7 in [1]}}{=\!=} \begin{bmatrix} \text{prox}_{\delta_{\mathcal{S}_{2.1}}}(\mathbf{z}_{1:m}) \\ \text{prox}_{\delta_{\mathcal{S}_{2.2}}}(\mathbf{z}_{m+1:m+n}) \\ \text{prox}_{\delta_{\mathcal{S}_{2.3}}}(\mathbf{z}_{m+n+1:m+2n}) \end{bmatrix} =$
  $\overset{\text{see th. 6.23 in [1]}}{=\!=} \begin{bmatrix} \text{Proj}_{\mathcal{S}_{2.1}}(\mathbf{z}_{1:m}) \\ \text{Proj}_{\mathcal{S}_{2.2}}(\mathbf{z}_{m+1:m+n}) \\ \text{Proj}_{\mathcal{S}_{2.3}}(\mathbf{z}_{m+n+1:m+2n}) \end{bmatrix} \overset{\text{see lemma 6.26 in [1]}}{=\!=} \begin{bmatrix} [\min(z_i, b_i)]_{i=1}^m \\ \left[\max(z_{m+i}, v_{1(i)})\right]_{i=1}^n \\ \left[\min(z_{m+n+i}, v_{2(i)})\right]_{i=1}^n \end{bmatrix}$
- As a termination criterion for the iterations (in order to detect convergence) we can use the following (with f_{opt} estimated by 'CVX' software):

$$f(\mathbf{x}) - f_{opt} \leq \epsilon$$

Finally, the DPG algorithm that solves problem (37) turns into:

Algorithm 20: DPG (primal representation) that solves problem (37).

Choose a random $\mathbf{y}_0 \in \mathbb{R}^m$

$L \leftarrow \frac{\|\mathbf{A}_0\|_2^2}{\sigma}$, $k \leftarrow 0$

while $(k = 0)$ *or* $(f(\mathbf{x}_k) - f_{opt} > \epsilon)$ **do**

$\mathbf{x}_k \leftarrow \mathbf{d} + \mathbf{A}_0^T \mathbf{y}_k$

$$\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k - \frac{1}{L}\mathbf{A}_0\mathbf{x}_k + \frac{1}{L}\begin{bmatrix} \left[\min\left((\mathbf{A}\mathbf{x}_k)_{(i)} - L\mathbf{y}_{k(i)}, b_i\right)\right]_{i=1}^{m} \\ \left[\max\left(\mathbf{x}_{k(i)} - L\mathbf{y}_{k(m+i)}, v_{1(i)}\right)\right]_{i=1}^{n} \\ \left[\min\left(\mathbf{x}_{k(i)} - L\mathbf{y}_{k(m+n+i)}, v_{2(i)}\right)\right]_{i=1}^{n} \end{bmatrix}$$

$k \leftarrow k + 1$

end

Figure 31 plots the quantity $f(\mathbf{x}_k) - f_{opt}$, while the algorithm converges to the solution of problem (37) (which is the projection of $\mathbf{d}$ onto $\mathcal{S}_3$). f_{opt} was obtained by solving the same problem with 'CVX' software. Furthermore, matrices $\mathbf{A}, \mathbf{b}, \mathbf{d}, \mathbf{v}_1, \mathbf{v}_2$ are the same with the respective matrices from section 12.*a*. Finally, the parameter ϵ was set to be equal with 10^{-9} again.

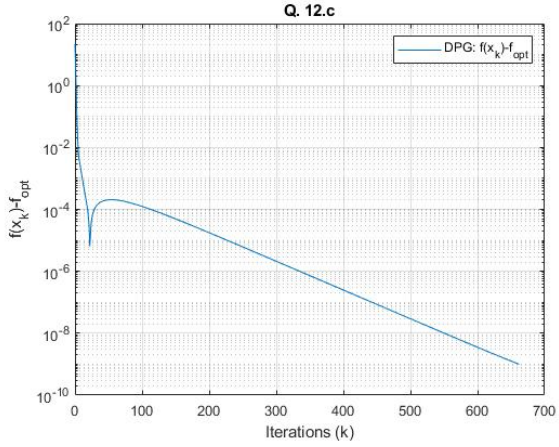


Figure 31: DPG for obtaining the projection of a vector onto $\mathcal{S}_3$.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex12.m`.

Exercise 13

Question 13.a. Let's consider a signal $\mathbf{d}_{true} \in \mathbb{R}^n$ and its noisy version $\mathbf{d} = \mathbf{d}_{true} + \mathbf{e}$, $\mathbf{e} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}_n\right)$. The total variation denoising problem is the problem of recovering the original signal $\mathbf{d}_{true}$ and can be defined as:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + R\left(\mathcal{A}\left(\mathbf{x}\right)\right) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 + R\left(\mathcal{A}\left(\mathbf{x}\right)\right)$$

where R is a regularization function and $\mathcal{A}\left(\mathbf{x}\right) = \mathbf{D}\mathbf{x} = \begin{bmatrix} x1 - x2 \\ \vdots \\ x_{n-1} - x_n \end{bmatrix} \in \mathbb{R}^{n-1}$ is a linear transformation of $\mathbf{x}$.

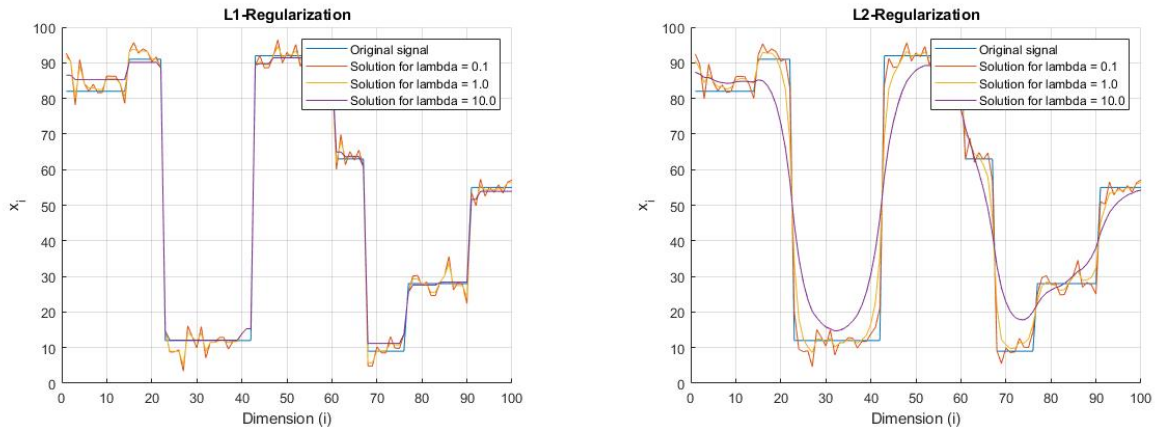
On the ℓ_1-regularized version of the total denoising problem we define $R(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ $(\lambda > 0)$:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + R\left(\mathcal{A}\left(\mathbf{x}\right)\right) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 + \lambda\|\mathbf{D}\mathbf{x}\|_1 \tag{38}$$

On the ℓ_2-regularized version of the total denoising problem we define $R(\mathbf{x}) = \lambda\|\mathbf{x}\|_2^2$, $(\lambda > 0)$:

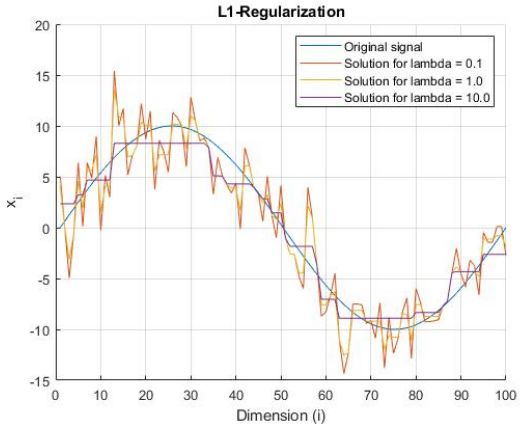
$$\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + R\left(\mathcal{A}\left(\mathbf{x}\right)\right) = \frac{1}{2}\|\mathbf{x} - \mathbf{d}\|_2^2 + \lambda\|\mathbf{D}\mathbf{x}\|_2^2 \tag{39}$$

Figures 32, 33 demonstrate an original (noiseless) signal $\mathbf{d}_{true}$, as well as its approximations that occur by solving problems (38), (39) respectively, for various λ. $\mathbf{d}_{true}$ is constructed to be either a piecewise vector, or a smooth (sinusoidal-shaped) vector with its maximum value being 10 and its minimum 0. The variance of the Gaussian noise has been set to $\sigma^2 = 9$. Moreover, the dimension of the signal has been set to $n = 100$.

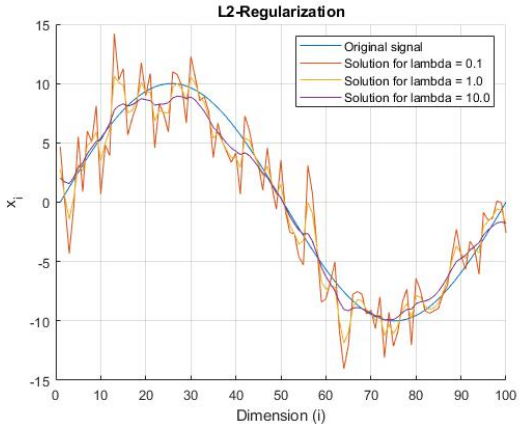


(a) Piecewise signal, ℓ_1-regularization. (b) Piecewise signal, ℓ_2-regularization.

Figure 32: Solutions of the opt. problems (38), (39) by 'CVX'



(a) Sinusoidal signal, ℓ_1-regularization.



(b) Sinusoidal signal, ℓ_2-regularization.

Figure 33: Solutions of the opt. problems (38), (39) by 'CVX'

We can easily observe some characteristics of the ℓ_1, ℓ_2 regularizers. λ parameter defines the amount of noise allowed in the solution of the problem. As a result, the denoising process for small λ produces more stable outputs compared to the same process for large λ. Also, ℓ_2 regularization is more suitable for smooth signals such as the sinusoidal one, while ℓ_1 regularization is better for constant signals, such as the piecewise one.

Question 13.b. On this section we will solve the ℓ_1-regularized version of the total denoising problem (see problem (38)) with both the Dual Proximal Gradient (DPG) and the Fast Dual Proximal Gradient (FDPG) algorithms. More in detail, if we set $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, then problem (38) can be rewritten as:

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{D}\mathbf{x}) = \frac{1}{2}\|\mathbf{x}-\mathbf{d}\|_2^2 + \lambda\|\mathbf{D}\mathbf{x}\|_1 \tag{40}$$

All requirements of the DPG, FDPG algorithms are satisfied, since f is proper closed and σ-strongly convex with $\sigma = 1$ (see example 5.19 in [1]). Also, g is proper closed and convex and $\mathbf{D}\mathbf{x}$ is a linear transformation of $\mathbf{x}$. Finally, the domains of f, g are $\mathbb{R}^n, \mathbb{R}^{n-1}$ respectively, thus for any $\mathbf{x}\in\mathbb{R}^n \Rightarrow \mathbf{D}\mathbf{x}\in\mathbb{R}^{n-1}$.

The general form of the DPG algorithm is given in section 12.*a*. For our problem, the algorithm is adapted as follows:

- $\mathcal{A}(\mathbf{x}_k) = \mathbf{D}\mathbf{x}_k = \mathbf{x}_{1:n-1} - \mathbf{x}_{2:n}$, $\mathcal{A}^T(\mathbf{y}_k) = \mathbf{D}^T\mathbf{y}_k = \begin{bmatrix}\mathbf{y}\\0\end{bmatrix} - \begin{bmatrix}0\\\mathbf{y}\end{bmatrix}$, $\|\mathcal{D}\|_2^2 = \|\mathbf{D}\|_2^2$
- $\text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\langle\mathbf{x},\mathcal{A}^T(\mathbf{y}_k)\rangle - f(\mathbf{x})\right\} = \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{x}^T\mathbf{D}^T\mathbf{y}_k - \frac{1}{2}\|\mathbf{x}-\mathbf{d}\|_2^2\right\} =$
  $= \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{\mathbf{x}^T\mathbf{D}^T\mathbf{y}_k - \frac{1}{2}\mathbf{x}^T\mathbf{x} + \mathbf{x}^T\mathbf{d} - \frac{1}{2}\mathbf{d}^T\mathbf{d}\right\} =$
  $= \text{argmax}_{\mathbf{x}\in\text{dom}f}\left\{-\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d}+\mathbf{D}^T\mathbf{y}_k\right)\right\|_2^2\right\} = \text{argmin}_{\mathbf{x}\in\text{dom}f}\left\{\frac{1}{2}\left\|\mathbf{x} - \left(\mathbf{d}+\mathbf{D}^T\mathbf{y}_k\right)\right\|_2^2\right\} =$
  $= \mathbf{d} + \mathbf{D}^T\mathbf{y}_k$
- $\text{prox}_{Lg}(\mathbf{z}) = \text{prox}_{L\lambda\|.\|_1}(\mathbf{z}) \overset{\text{see example 6.8 in [1]}}{=\!=\!=} \mathcal{T}_{L\lambda}(\mathbf{z})$ where the soft thresholding operator $\mathcal{T}_\lambda$ is explained in section 6.*b*.

- As a termination criterion for the iterations (in order to detect convergence) we can use the following (with $f_{opt} = f(\mathbf{x}_{opt})$ estimated by the 'CVX' software as shown in section 13.*a*):

$$f(\mathbf{x}) - f_{opt} \leq \epsilon$$

Finally, the DPG algorithm that solves problem (40) turns into:

Algorithm 21: DPG (primal representation) that solves problem (40).

Choose a random $\mathbf{y}_0 \in \mathbb{R}^{n-1}$
$L \leftarrow \frac{\|\mathbf{D}\|_2^2}{\sigma}$, $k \leftarrow 0$
while $(k = 0)$ *or* $(f(\mathbf{x}_k) - f_{opt} > \epsilon)$ **do**
  $\mathbf{x}_k \leftarrow \mathbf{d} + \mathbf{D}^T \mathbf{y}_k$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{y}_k - \frac{1}{L}\mathbf{D}\mathbf{x}_k + \frac{1}{L}\mathcal{T}_{L\lambda}\left(\mathbf{D}\mathbf{x}_k - L\mathbf{y}_k\right)$
  $k \leftarrow k + 1$
end

FDPG is the accelerated version of DPG and its general form is:

Algorithm 22: FDPG algorithm - primal representation.

Choose a random $\mathbf{y}_0 \in \text{dom}g$ and a parameter $L \geq \frac{\|\mathcal{A}\|_2^2}{\sigma}$
$\mathbf{w}_0 \leftarrow \mathbf{y}_0$, $t_0 \leftarrow 1$, $k \leftarrow 0$
while *the iterations haven't converged* **do**
  $\mathbf{u}_k \leftarrow \text{argmax}_{\mathbf{u} \in \text{dom}f} \left\{ \langle \mathbf{u}, \mathcal{A}^T(\mathbf{w}_k) \rangle - f(\mathbf{u}) \right\}$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{w}_k - \frac{1}{L}\mathcal{A}(\mathbf{u}_k) + \frac{1}{L}\text{prox}_{Lg}\left(\mathcal{A}(\mathbf{u}_k) - L\mathbf{w}_k\right)$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{w}_{k+1} \leftarrow \mathbf{y}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{y}_{k+1} - \mathbf{y}_k)$
  $\mathbf{x}_{k+1} \leftarrow \text{argmax}_{\mathbf{x} \in \text{dom}f} \left\{ \langle \mathbf{x}, \mathcal{A}^T(\mathbf{y}_k) \rangle - f(\mathbf{x}) \right\}$
  $k \leftarrow k + 1$
end

If we adapt FDPG to solve problem (40) (and according to the observations that were made previously), it turns into:

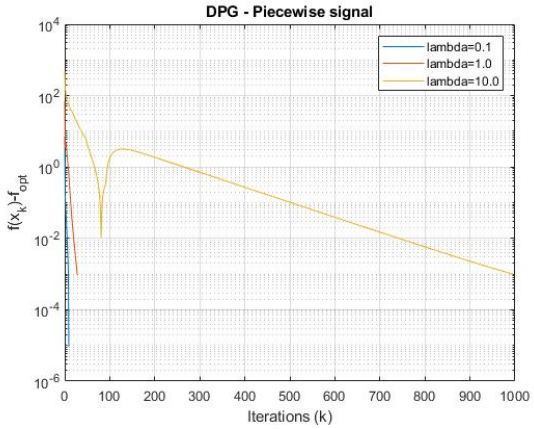
Algorithm 23: FDPG algorithm (primal representation) that solves problem (40).

Choose a random $\mathbf{y}_0 \in \mathbb{R}^{n-1}$
$L \leftarrow \frac{\|\mathbf{D}\|_2^2}{\sigma}$, $\mathbf{w}_0 \leftarrow \mathbf{y}_0$, $t_0 \leftarrow 1$, $k \leftarrow 0$
while $(k = 0)$ *or* $(f(\mathbf{x}_k) - f_{opt} > \epsilon)$ **do**
  $\mathbf{u}_k \leftarrow \mathbf{d} + \mathbf{D}^T \mathbf{w}_k$
  $\mathbf{y}_{k+1} \leftarrow \mathbf{w}_k - \frac{1}{L}\mathbf{D}\mathbf{u}_k + \frac{1}{L}\mathcal{T}_{L\lambda}\left(\mathbf{D}\mathbf{u}_k - L\mathbf{w}_k\right)$
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$
  $\mathbf{w}_{k+1} \leftarrow \mathbf{y}_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{y}_{k+1} - \mathbf{y}_k)$
  $\mathbf{x}_{k+1} \leftarrow \mathbf{d} + \mathbf{D}^T \mathbf{y}_{k+1}$
  $k \leftarrow k + 1$
end

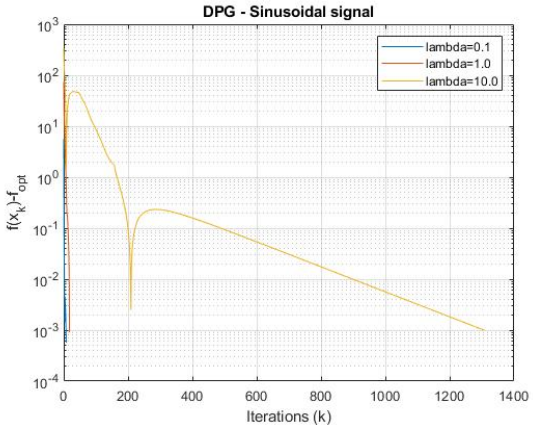
Figures 34, 35 demonstrate the performance of DPG, FDPG algorithms while they solve problem (40), for various λ. More specifically, they plot quantity $f(\mathbf{x}_k) - f_{opt}$

versus the number of iterations k. The parameter of the termination criterion has been set to $\epsilon = 10^{-3}$. We examine the same noisy signals as in section 13.*a*.

As expected, the FDPG is remarkably faster than the DPG technique. Moreover, the convergence is slower, when λ increases.

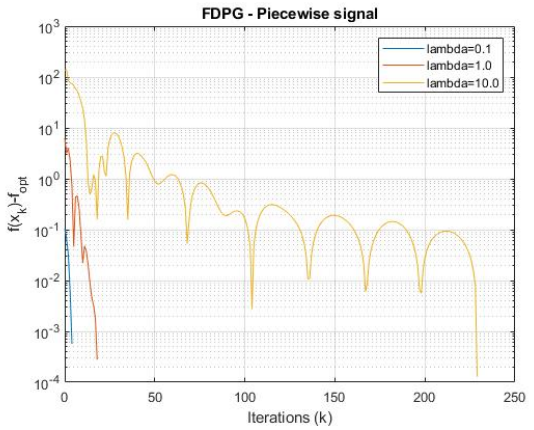


(a) Piecewise signal, ℓ_1-regularization.

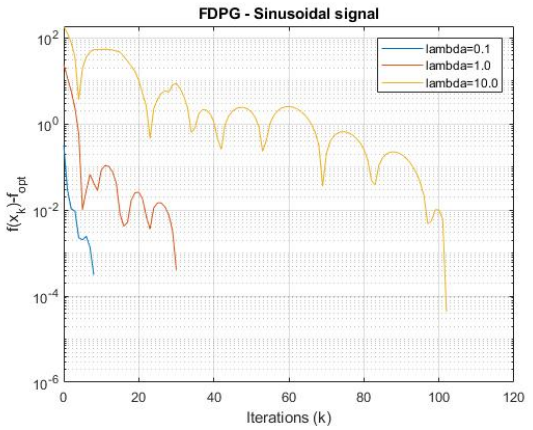


(b) Piecewise signal, ℓ_2-regularization.

Figure 34: Solutions of the opt. problem (40) with the DPG algorithm.



(a) Sinusoidal signal, ℓ_1-regularization.



(b) Sinusoidal signal, ℓ_2-regularization.

Figure 35: Solutions of the opt. problemm (40) with the FDPG algorithm.

The MATLAB code that was developed for the purposes of this exercise can be found in file `convexopt_ex13.m`.

References

[1] A. Beck, First-Order Methods in Optimization, SIAM, 2017