ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Electrical and Computer Engineering

Advanced Topics in Convex Optimization

Exercises Report

Instructor: Athanasios P. Liavas

Ioannis-Leonidas Steiakakis ID: – – –

Contents

Every Section, Theorem, Example or Corollary reference is from the book "First-Order Methods in Optimization" from Amir Beck.

1 Exercise 1

Implement the Corollary 6.29 algorithm to find the projection of a given point onto the unit simplex.

Here are two plots for the two-dimensional case ($n = 2$):

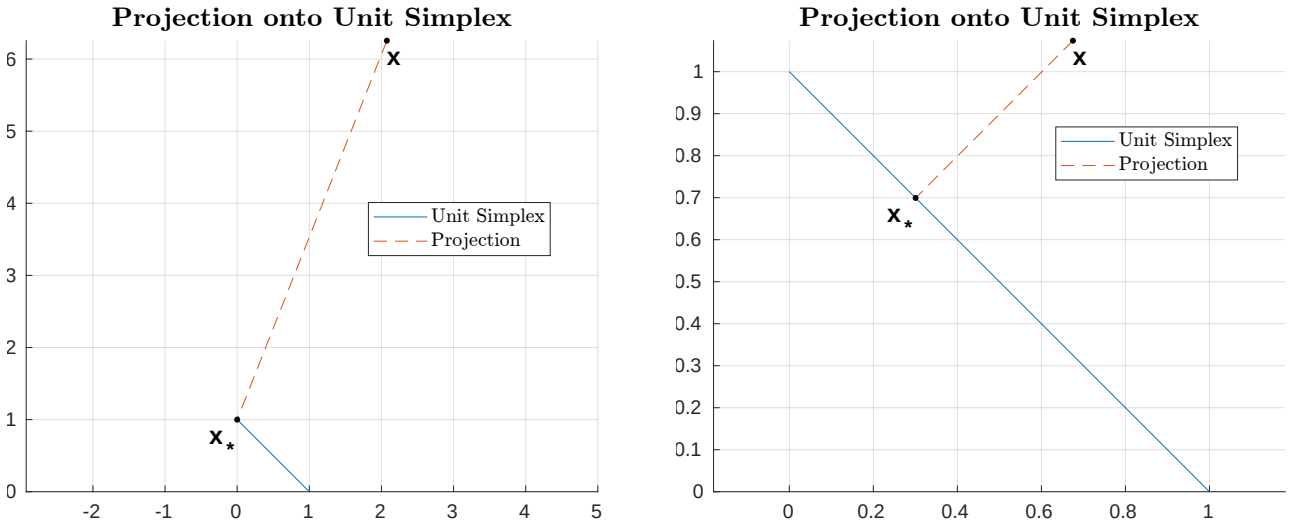

Figure 1

What is happening here is that the projection is trying to find the shortest path to the unit simplex, as expected.

2 Exercise 2

Implement the Algorithm 1 from the Example 8.23.

During the problem generation, $\mathbf{A}$ and $\mathbf{b}$ are generated in such a way that the problem is solvable, i.e. $S \neq \emptyset$. In particular, firstly $\mathbf{A}$, $\mathbf{x}^*$ are generated randomly, and then $\mathbf{b} = \mathbf{A}\mathbf{x}^*$; then we are trying to find x^*.

The solution is in the MATLAB file `Ex2.m`.

3 Exercise 3

At first, let's express the problem in the standard form

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m \\ & h_i(\mathbf{x}) = 0, \quad i = 1, \dots, p, \end{aligned}$$

so, the requested problem can be expressed equivalently

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & -x_i \leq 0, \qquad i = 1, \dots, n \\ & \mathbf{e}^T\mathbf{x} - 1 = 0, \end{aligned}$$

where $\mathbf{e} \in \mathbb{R}^n$ is a vector whose every element is 1.

The functions of the standard form for this problem are

$$\begin{aligned} f_0(\mathbf{x}) &:= \mathbf{c}^T\mathbf{x} \\ f_i(\mathbf{x}) &:= -x_i = -\mathbf{e}_i^T\mathbf{x}, \quad i = 1, \dots, n \\ h_1(\mathbf{x}) &:= \mathbf{e}^T\mathbf{x} - 1, \end{aligned}$$

where $\mathbf{e}_i$ is the i-th column of the identity matrix in $\mathbb{R}^{n\times n}$.

The KKT conditions of this problem are

$$\underbrace{\nabla f_0(\mathbf{x})}_{\mathbf{c}} + \sum_{i=1}^{n} u_i \underbrace{\nabla f_i(\mathbf{x})}_{-\mathbf{e}_i} + \underbrace{\mathbf{A}^T}_{\mathbf{e}}\mathbf{v} = \mathbf{0} \iff \boxed{\mathbf{c} - \mathbf{u} + \mathbf{e}v = \mathbf{0}} \tag{3.1}$$

$$\boxed{\mathbf{u} \geq \mathbf{0}} \tag{3.2}$$

$$u_i f_i(\mathbf{x}) = 0, \quad i = 1, \dots, n \iff \boxed{u_i x_i = 0, \quad i = 1, \dots, n} \tag{3.3}$$

$$\boxed{\mathbf{x} \geq \mathbf{0}} \tag{3.4}$$

$$\boxed{\mathbf{e}^T\mathbf{x} = 1} \tag{3.5}$$

What is going to be needed for the next step is that

$$\exists\, j \in \{1, \dots, n\} \text{ such that } u_j = 0. \tag{3.6}$$

The proposition above stands because if it did not, then $u_j \neq 0$ for all j, so by (3.3) it occurs that $x_j = 0$ for all j, i.e. $\mathbf{x} = \mathbf{0}$, which contradicts the equation (3.5).

From now on, j will be the element mentioned in (3.6), i.e. $u_j = 0$.

Solving the j-th equation from (3.1) yields

$$\begin{aligned} & c_j - u_j + v = 0 \\ \iff & v = -c_j, \end{aligned}$$

since $u_j = 0$.

Now, the equation (3.1) becomes

$$\mathbf{u} = \mathbf{c} - c_j \mathbf{e}. \tag{3.7}$$

Using (3.2), (3.7) we have

$$\begin{aligned} & \mathbf{c} - c_j \mathbf{e} \geq \mathbf{0} \\ \iff & c_i \geq c_j \quad \forall\, i \in \{1, \ldots, n\} \\ \iff & c_j = \min_{i \in \{1,\ldots,n\}} c_i, \end{aligned}$$

The value of c_j is now known and from now on it will be noted as $c_{\min}$. So, the equation (3.7) can be rewritten as

$$\mathbf{u} = \mathbf{c} - c_{\min} \mathbf{e}.$$

From the equation (3.3) occurs that if $u_i \neq 0 \iff c_i \neq c_{\min}$ then $x_i = 0$; else, x_i can take whichever value such that the equations (3.4), (3.5) hold. Thus, the final result is

$$\boxed{\mathbf{x} \in \left\{ \mathbf{r} \in \mathbb{R}^n \;\middle|\; \forall\, i \in I_{\neq} : r_i = 0, \;\; \mathbf{r} \geq \mathbf{0}, \;\; \sum_{i=1}^{n} r_i = 1 \right\}}$$

where

$$\begin{aligned} I_{\neq} &= \{i \in \{1, \ldots, n\} : u_i \neq 0\} \\ &= \{i \in \{1, \ldots, n\} : c_i \neq c_{\min}\}. \end{aligned}$$

A weak result is

$$\begin{aligned} x_i &= \begin{cases} 0 & \text{if } \; i \in I_{\neq} \\ \dfrac{1}{n - |I_{\neq}|} & \text{else} \end{cases}, \qquad i = 1, \ldots, n \\ \iff \mathbf{x} &= \frac{1}{n - |I_{\neq}|} (\mathbf{e} - |\operatorname{sgn}(\mathbf{u})|), \end{aligned}$$

where

$$|I_{\neq}| = \sum_{i=1}^{n} |\mathrm{sgn}(u_i)|.$$

Another weak result is

$$\mathbf{x} = \mathbf{e}_i\,, \quad i \in I_0 = \{1, \ldots, n\} \setminus I_{\neq}.$$

4 Exercise 4

4.1 Question a

The problem

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

is solved via CVX.

4.2 Question b

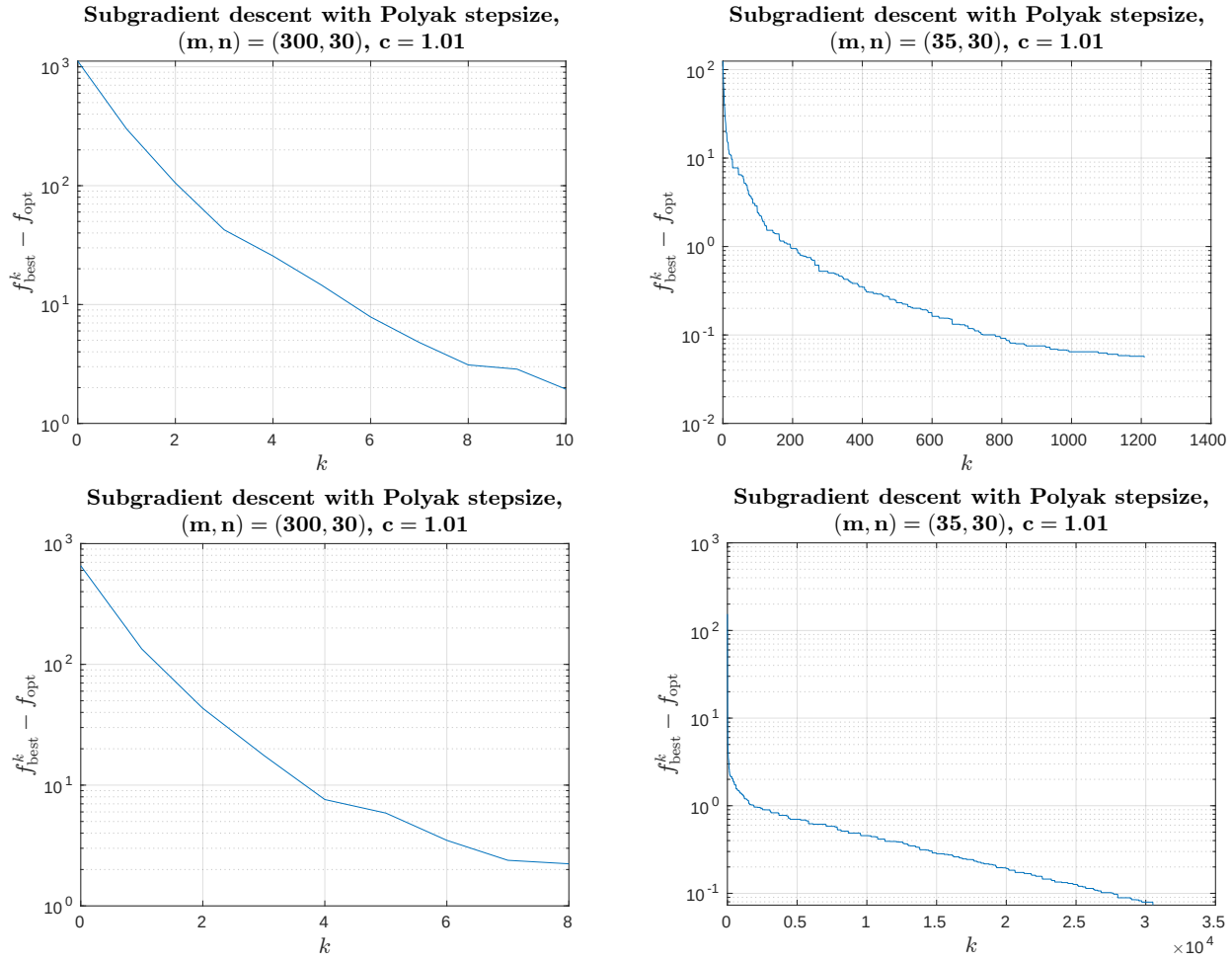

Figure 2: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated using Gaussian distribution; in the second row, using uniform distribution.

4.3 Question c

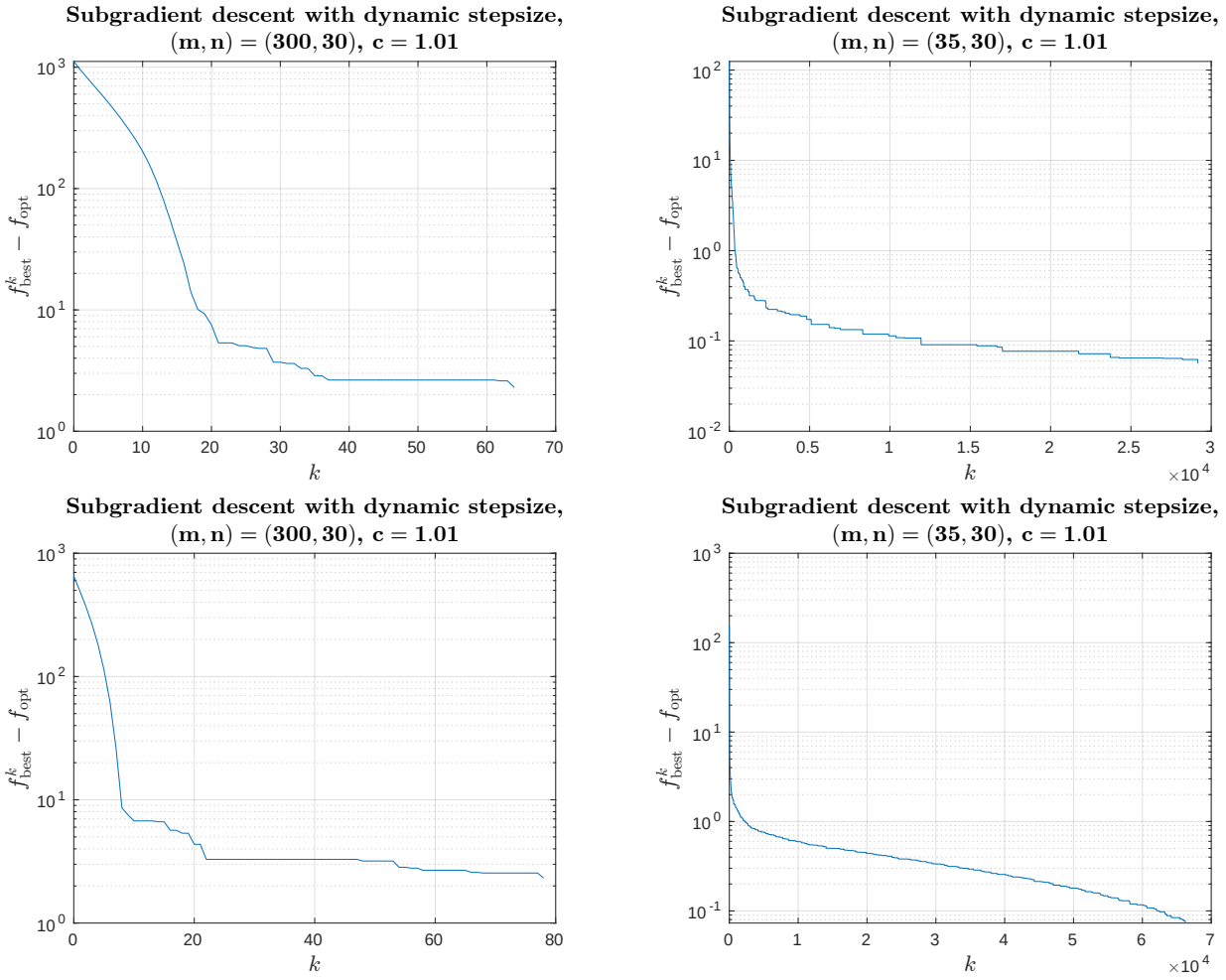

Figure 3: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

4.4 Question d

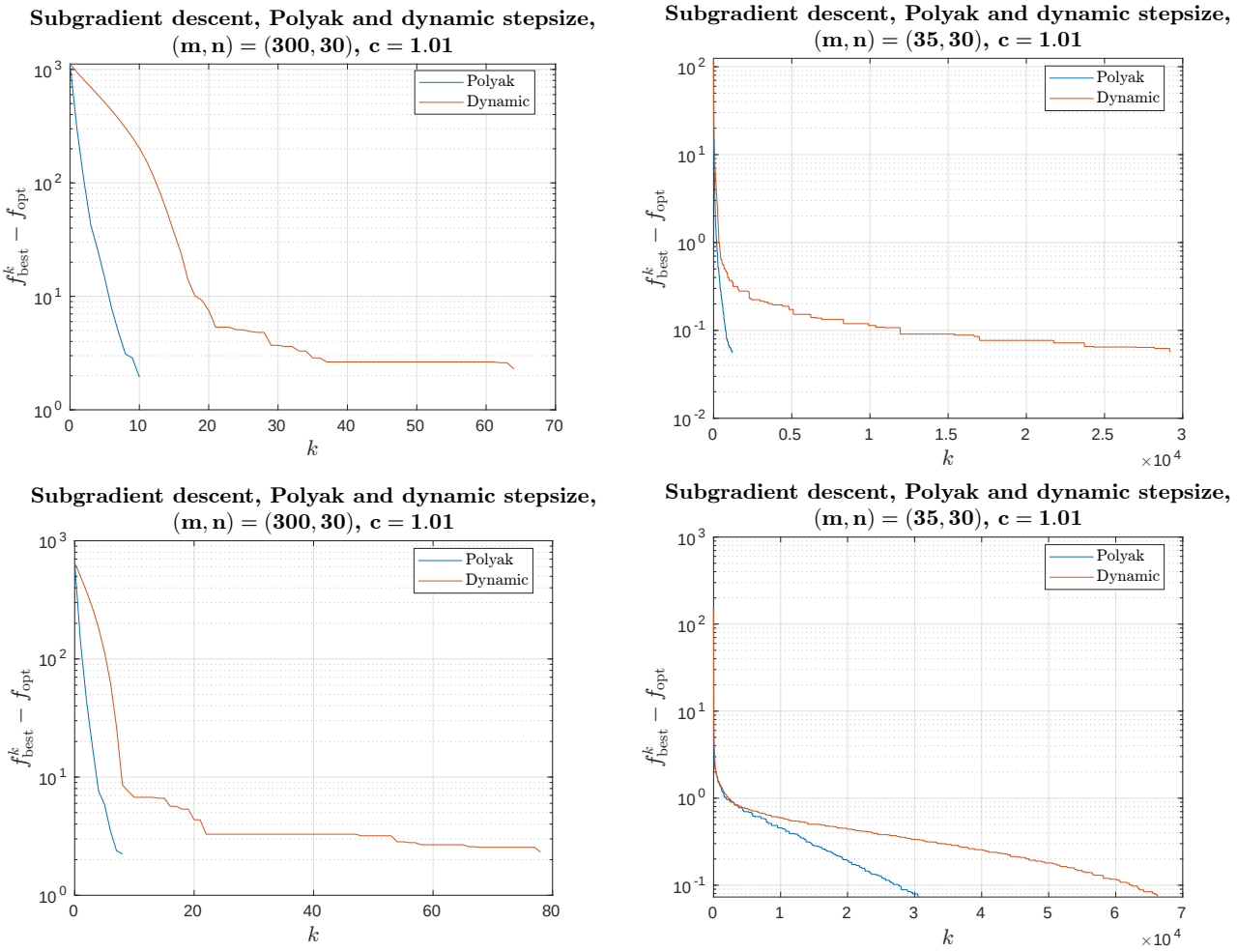

Figure 4: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

Similar patterns are observed among the distributions used to generate the problem. However, it is a lot easier for the problem to be solved if $m \gg n$ instead of $m \gtrapprox n$, as is evident from the number of iterations. In every case, the Polyak stepsize converges much faster than the dynamic stepsize, although there might be some problems where both stepsizes converge at similar rate.

$m \gg n$ means that the system is extremely overdetermined (more equations than unknowns).The more equations there are, the more the potential solutions are narrowed down, making it easier to find the solution.

4.5 Question e

We want to compute a value for L_f such that $\|\mathbf{g}\|_2 \leq L_f \;\; \forall\, \mathbf{g} \in \partial f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n$.

Let $h : \mathbb{R}^n \mapsto \mathbb{R}$ with $h(\mathbf{x}) = \|\mathbf{x}\|_1$.

Using the affine transformation rule of subdifferential calculus, we have that

$$\partial f(\mathbf{x}) = \mathbf{A}^T(\partial h(\mathbf{A}\mathbf{x} - \mathbf{b})),$$

so, $\mathbf{g} = \mathbf{A}^T\mathbf{v}$ where $\mathbf{v} \in \partial h(\mathbf{A}\mathbf{x} - \mathbf{b})$.

So,

$$\|\mathbf{g}\|_2 = \left\|\mathbf{A}^T\mathbf{v}\right\|_2 \leq \left\|\mathbf{A}^T\right\|_{2,2}\|\mathbf{v}\|_2 \leq \sigma_{\max}(\mathbf{A})\sqrt{m},$$

since

- $\left\|\mathbf{A}^T\right\|_{2,2} = \sigma_{\max}\left(\mathbf{A}^T\right) = \sigma_{\max}(\mathbf{A})$
- $\mathbf{v}$ is a subgradient of the ℓ_1-norm, so $|v_i| \leq 1$, so $\|\mathbf{v}\|_2 \leq \|\mathbf{e}\|_2 = \sqrt{m}$.

Hence, we can choose $L_f = \sigma_{\max}(\mathbf{A})\sqrt{m}$.

Below are the figures previously shown, now including Polyak's upper bound.

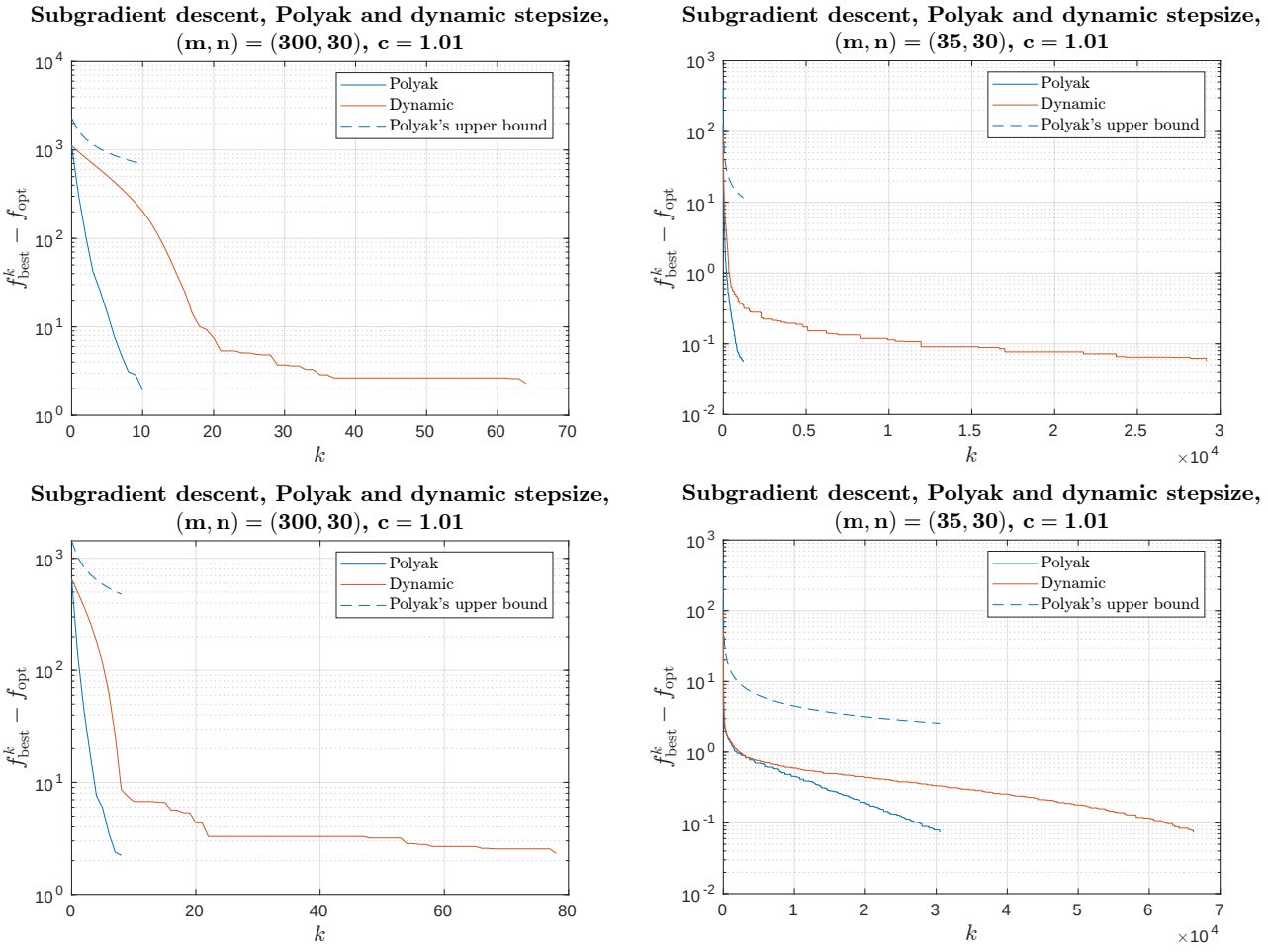

Figure 5: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

4.6 Question f

As stated in the Example 8.36, this problem has the form

$$\min\left\{ f(\mathbf{x}) \equiv \sum_{i=1}^{m} f_i(\mathbf{x}) \;\middle|\; \mathbf{x} \in C \right\},$$

where

$$\begin{aligned} f(\mathbf{x}) &:= \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 = \sum_{i=1}^{m} |(\mathbf{A}\mathbf{x} - \mathbf{b})_i| \\ f_i(\mathbf{x}) &:= |(\mathbf{A}\mathbf{x} - \mathbf{b})_i|, \quad i = 1, \ldots, m \\ C &= \mathbb{R}^n. \end{aligned}$$

At this point, it must be noted that C is not compact (because it is not bounded), as it is assumed in the Example 8.36. So, choose Θ arbitrarily as

$$\Theta = \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2.$$

where $\mathbf{x}^*$ is the optimal point found using CVX.

Also, $\forall\, \mathbf{x} \in \mathbb{R}^n : P_C(\mathbf{x}) = \mathbf{x}$, as $C = \mathbb{R}^n$.

Assume $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix}$, so

$$\begin{aligned} f_i(\mathbf{x}) &= |(\mathbf{A}\mathbf{x} - \mathbf{b})_i| \\ &= \left|\mathbf{a}_i^T\mathbf{x} - b_i\right|. \end{aligned}$$

So, using the result from question e, we have

$$L_{f_i} = \sigma_{\max}(\mathbf{a}_i),$$

which will be used in the algorithm to compute $\tilde{L}_f = \sqrt{m}\sqrt{\sum_{i=1}^{m} L_{f_i}^2}$.

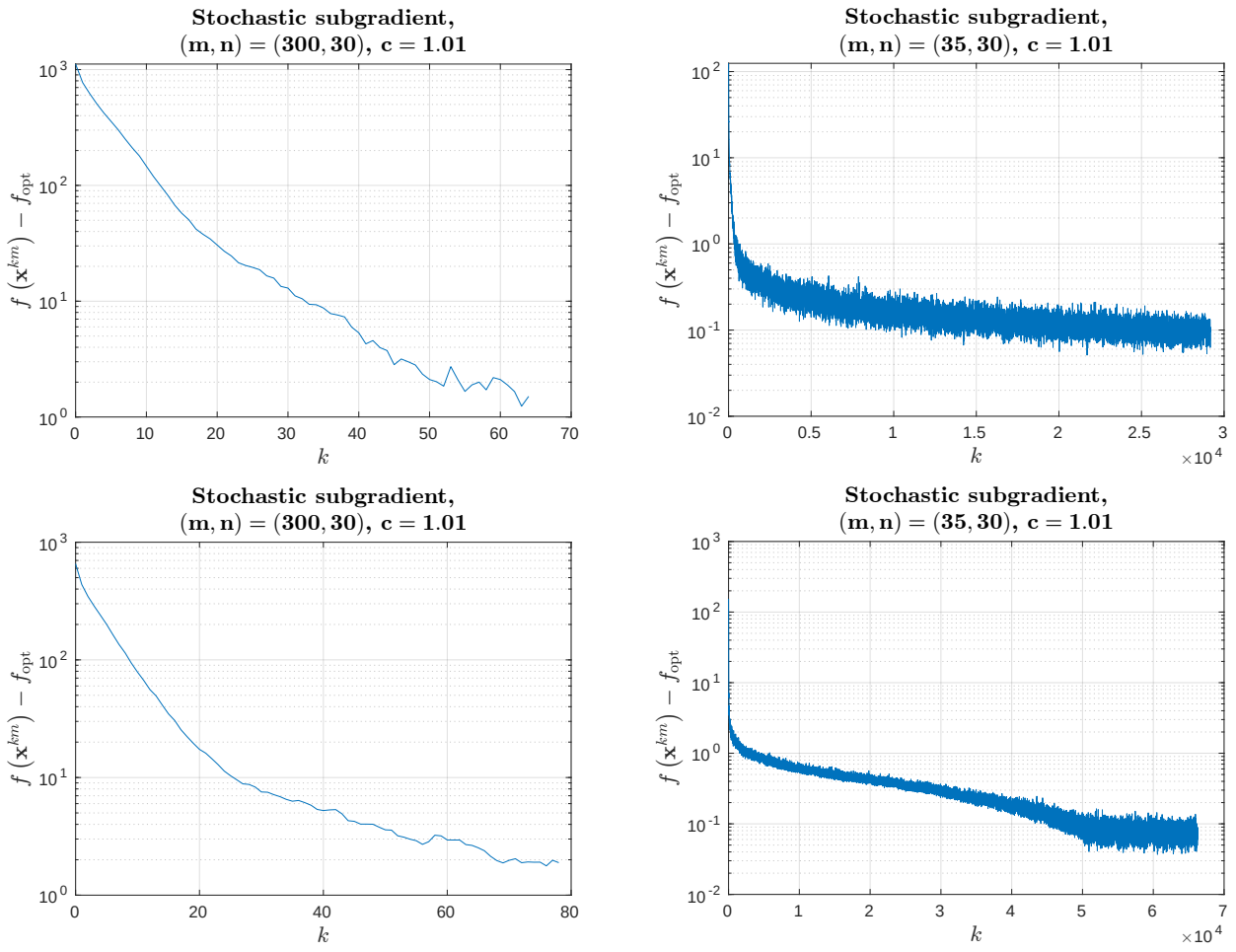

Figure 6: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

The same pattern is observed here as in question d: the algorithm converges much faster if $m \gg n$ instead of $m \gtrapprox n$.

4.7 Question g

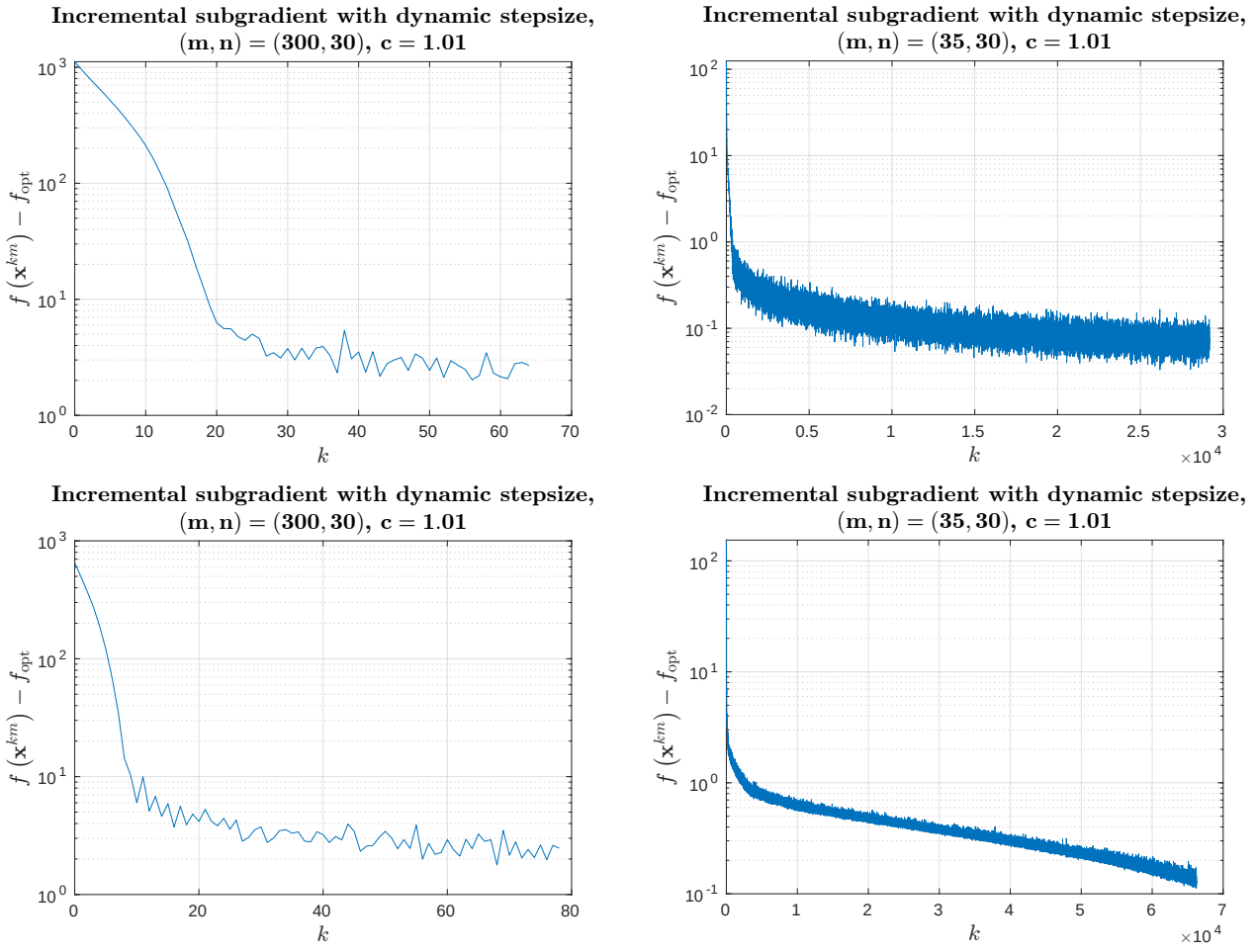

Figure 7: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

The same pattern is observed here as in the above question: the algorithm converges much faster if $m \gg n$ instead of $m \gtrapprox n$.

4.8 Question h

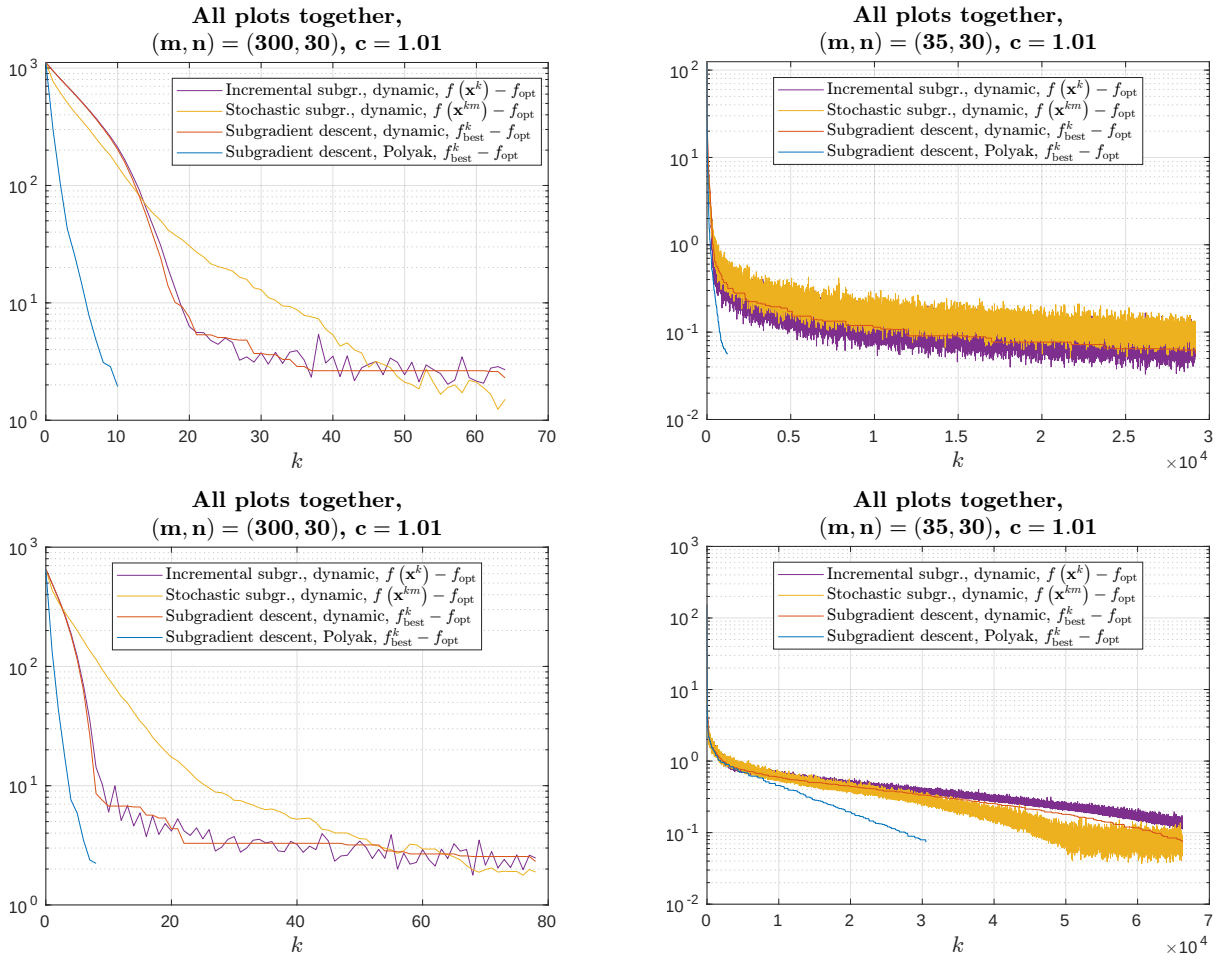

Figure 8: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

5 Exercise 5

5.1 Question a

The problem

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

is solved via CVX.

5.2 Question b

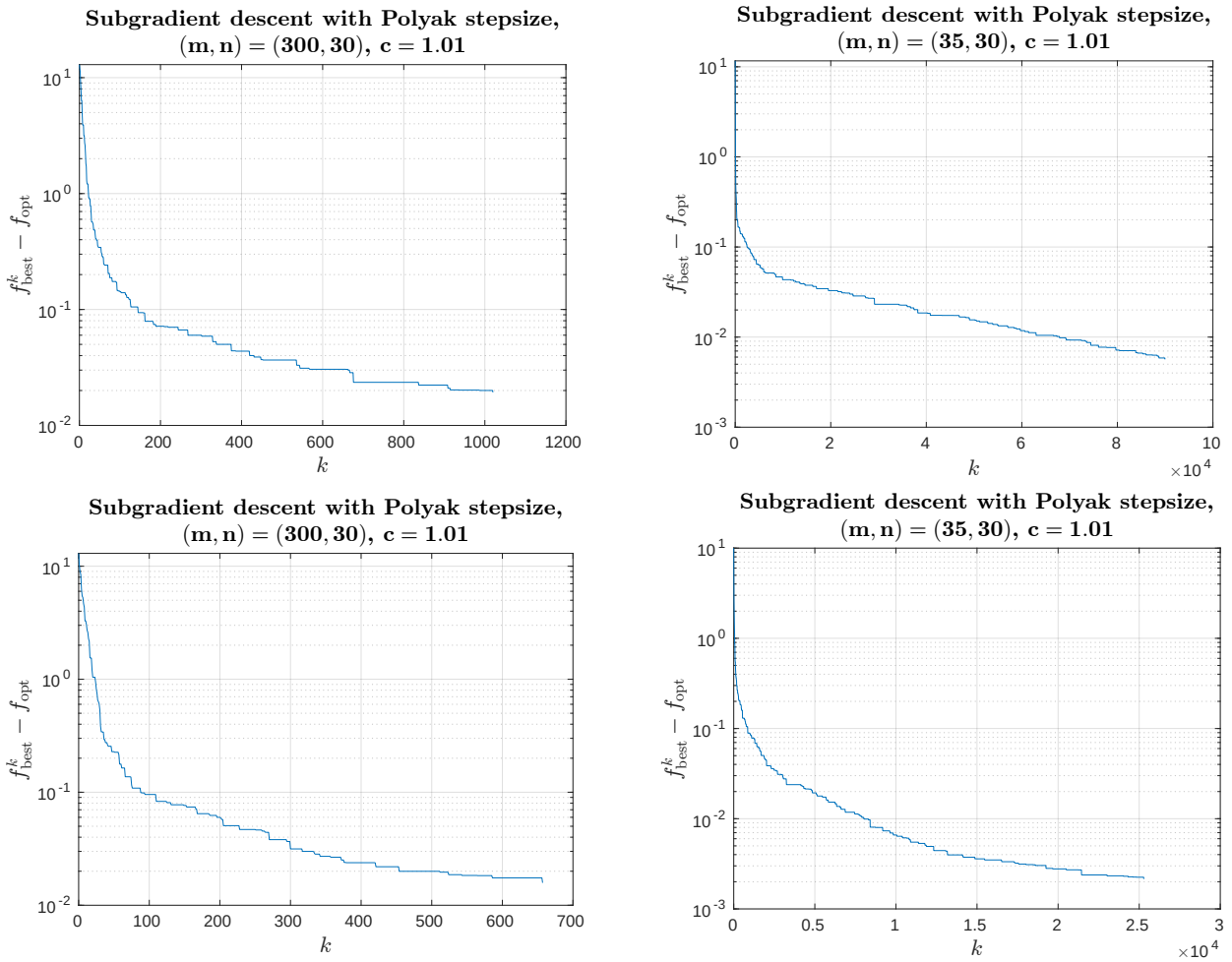

Figure 9: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

5.3 Question c

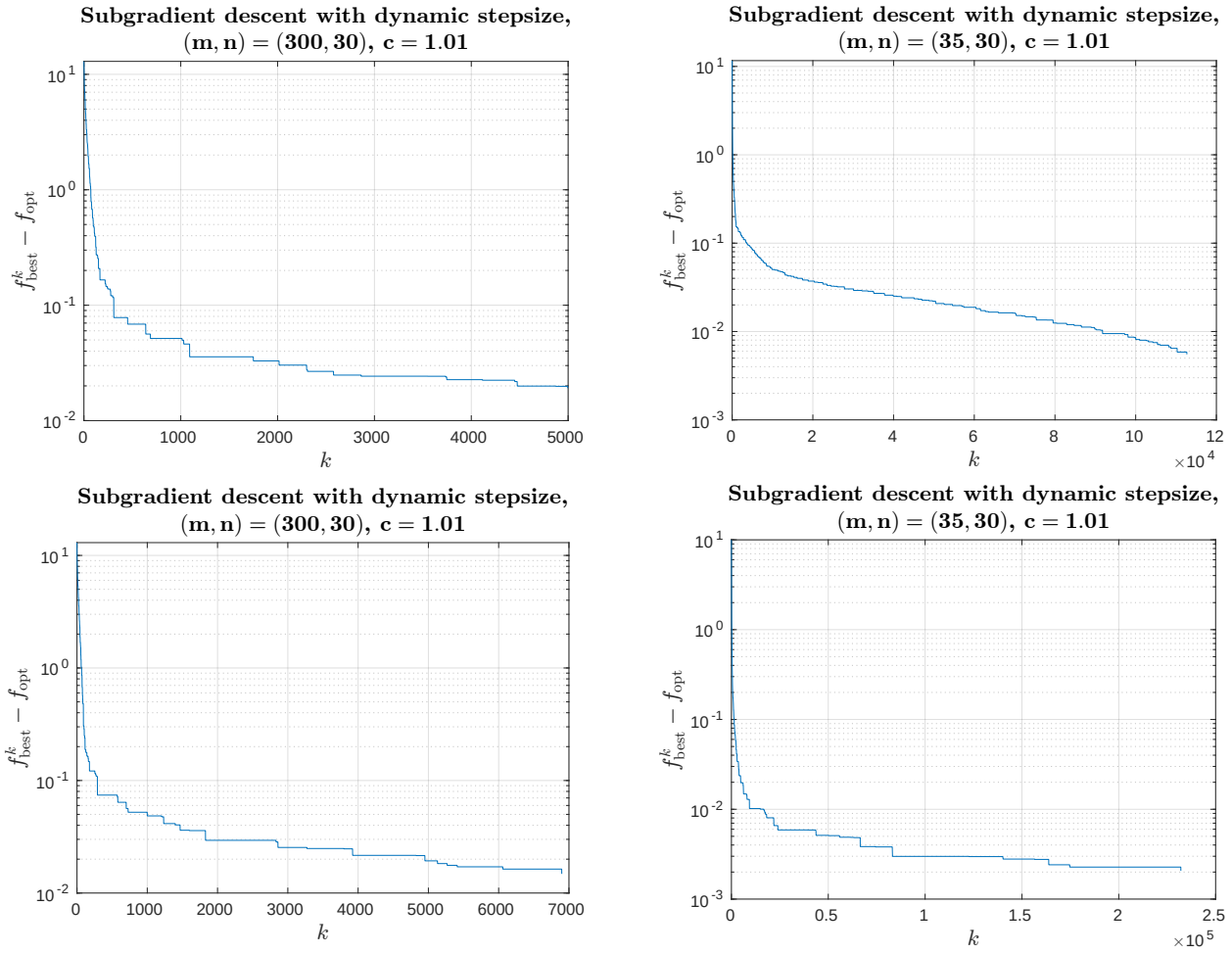

Figure 10: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

5.4 Question d

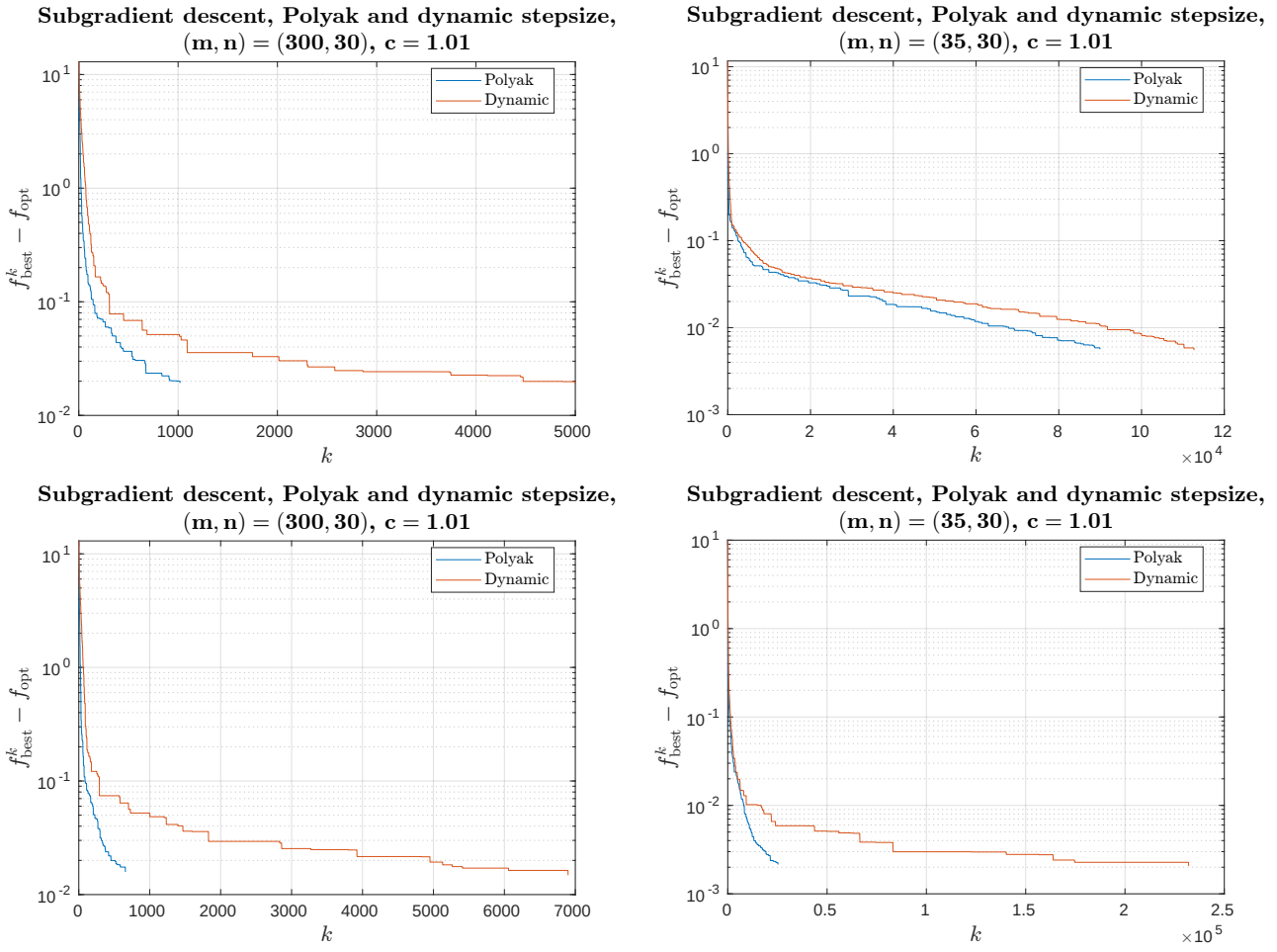

Figure 11: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

The same pattern is observed here as in exercise 4: the algorithm converges much faster if $m \gg n$ instead of $m \gtrapprox n$.

5.5 Question e

Following the exact same steps as in the question 4e, but with $h(\mathbf{x}) = \|\mathbf{x}\|_\infty$ (nothing else changes), we get the same result

$$\|\mathbf{g}\|_2 \leq \sigma_{\max}(\mathbf{A})\sqrt{m}.$$

Hence, we can choose $L_f = \sigma_{\max}(\mathbf{A})\sqrt{m}$.

Below are the figures previously shown, now including Polyak's upper bound.

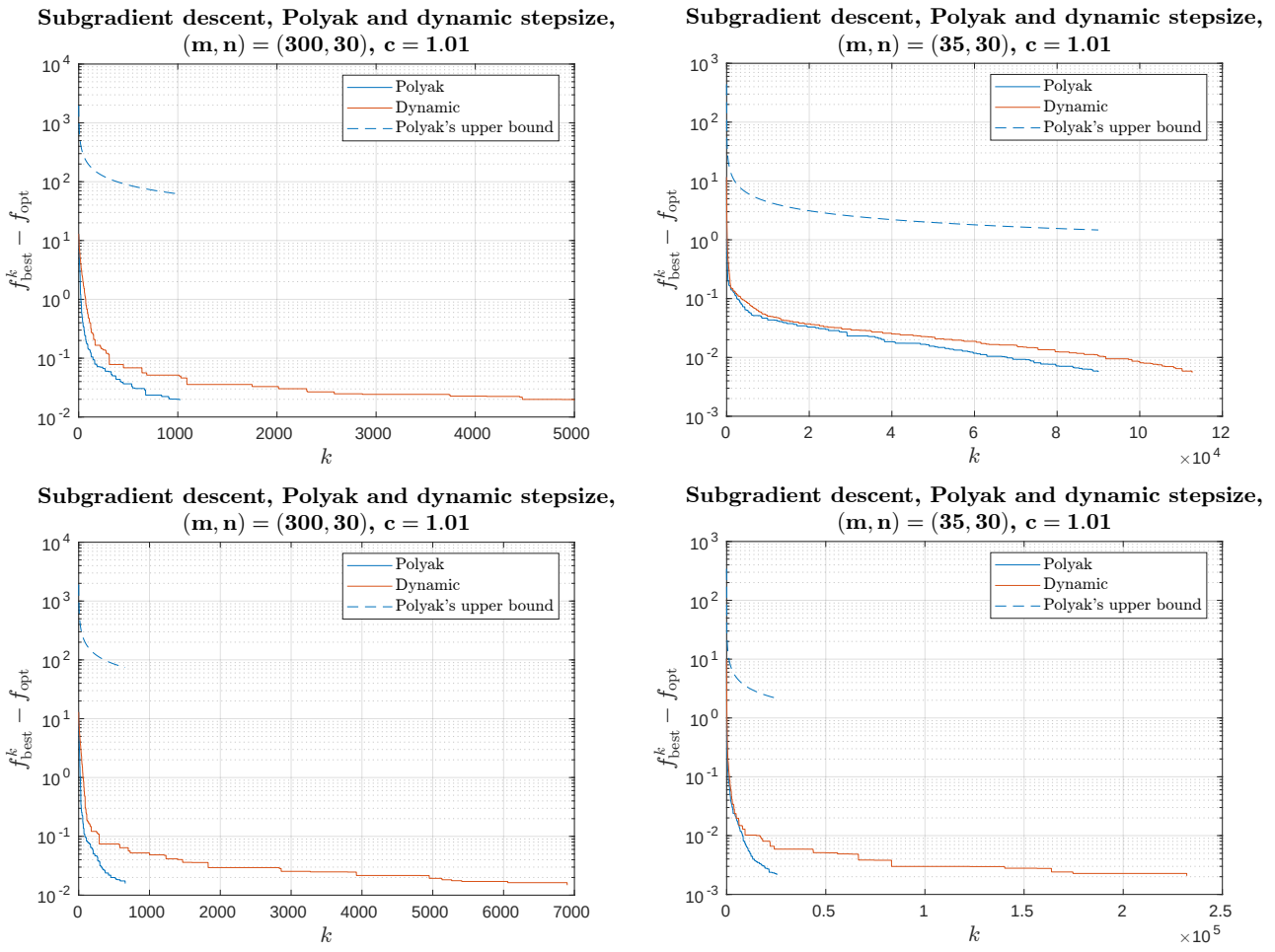

Figure 12: In the first row of figures, $\mathbf{A}, \mathbf{b}$ were generated via Gaussian distribution; in the second row, via uniform distribution.

6 Exercise 6

6.1 Question a

The problems

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f_1(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1 \ , \\ \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f_2(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2 \end{aligned}$$

are solved via CVX.

By solving those problems, we are trying to recover $\mathbf{x}_{\text{s}}$ which is a sparse vector.

Running the MATLAB script `Ex6.m`, the value $\|\mathbf{x}^* - \mathbf{x}_{\text{s}}\|_2$ is printed in the console, where $\mathbf{x}^*$ is the optimal point of either the $L1$ or the $L2$ regularized problem computed via CVX.

The difference between the solutions of the $L1$ and the $L2$ regularized problems is obvious, and it is dependent on λ.

In order to show the comparison of the $L1$ and the $L2$ regularized problems solutions for different values of λ, another script has been created which is named `Ex6_lambda_fine_tuning.m`. Running it for $m = 10, n = 50$ for 10 random generated problems, the following figure emerges:

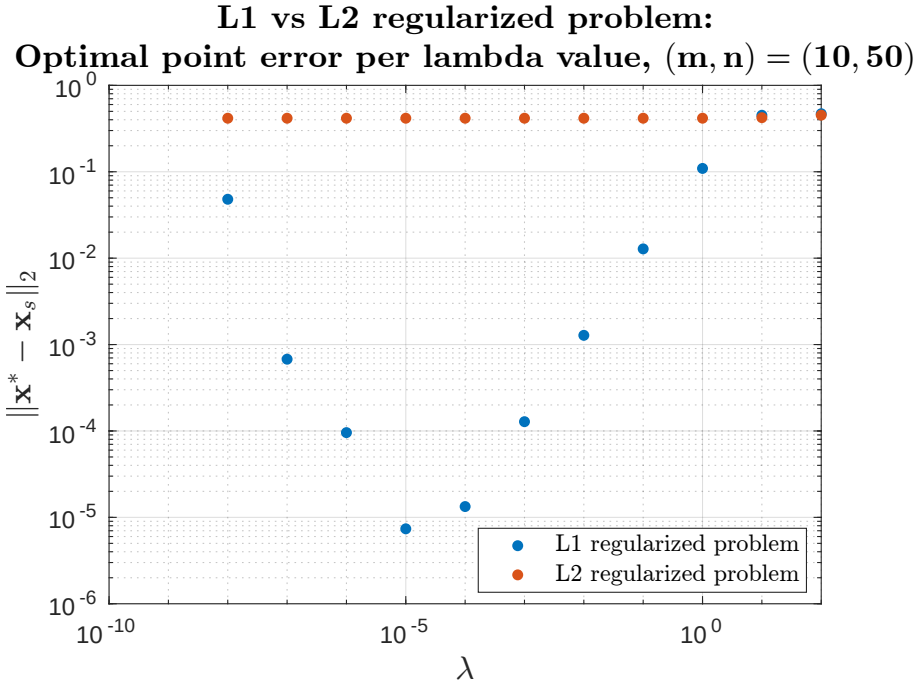

Figure 13

It is obvious that the $L1$ regularized problem provides much better solutions for $\mathbf{x}_s$.

This happens as $L1$ regularization promotes sparsity. So, even with such an underdetermined problem, using $L1$ regularization a very accurate result can be achieved.

6.2 Question b

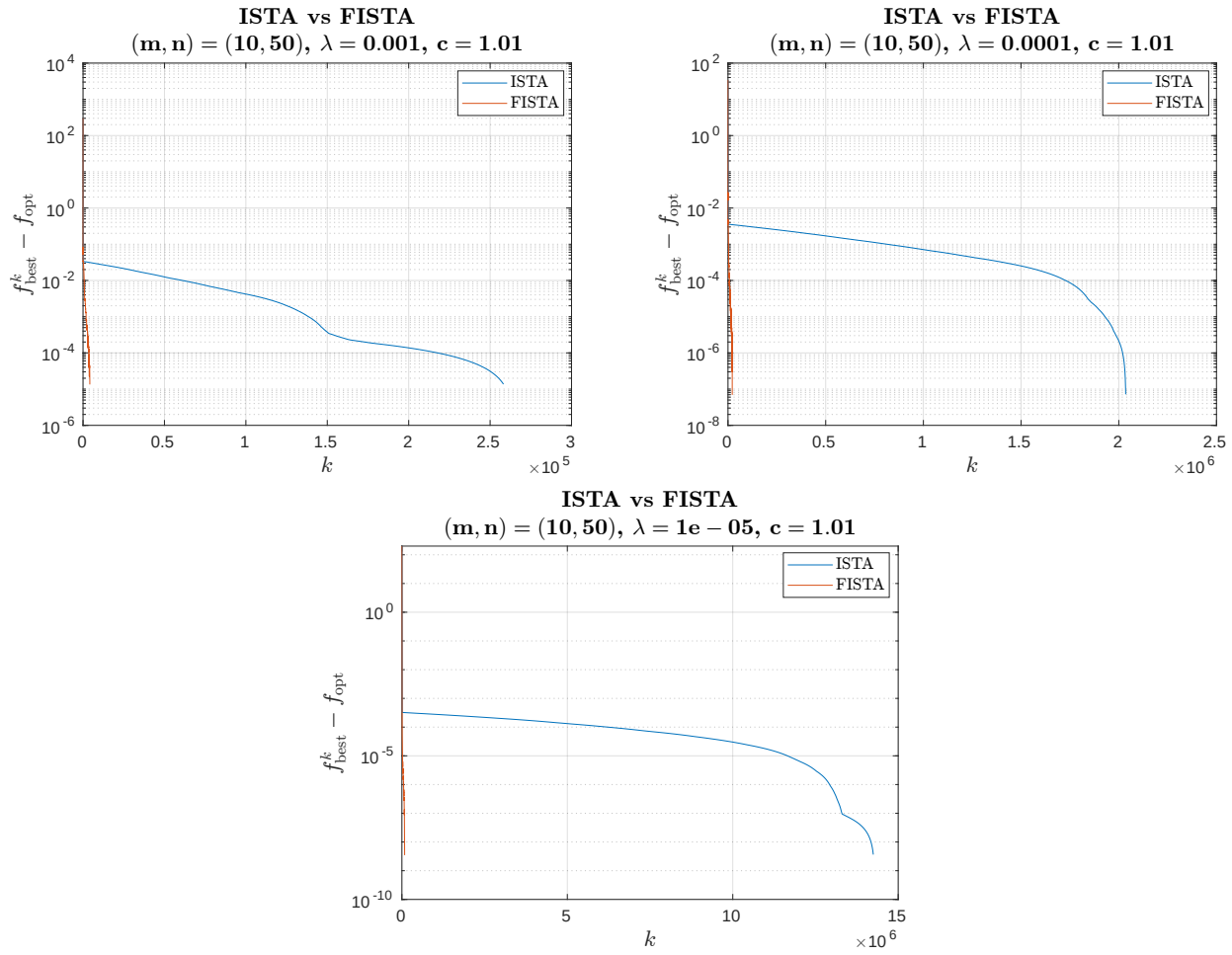

Figure 14

As demonstrated in the convergence plots above, the FISTA method converges significantly faster than the ISTA method.

Furthermore, it should be noted that the convergence rate of the algorithms is dependent on the value of the regularization parameter λ. From the plots, it appears that as the value of λ gets smaller, the more iterations are required.

Below are the console outputs for each one of the above plots:

- $\lambda = 10^{-3}$

```
Solving the L1 regularized problem (CVX)... ||x_star1 - x_s|| = 0.000135
Solving the L2 regularized problem (CVX)... ||x_star2 - x_s|| = 1.253383
Running ISTA on the L1 reg. problem...  Elapsed time is 2.341274 seconds.
Running FISTA on the L1 reg. problem... Elapsed time is 0.039781 seconds.
```

- $\lambda = 10^{-4}$

```
Solving the L1 regularized problem (CVX)... ||x_star1 - x_s|| = 0.000010
Solving the L2 regularized problem (CVX)... ||x_star2 - x_s|| = 0.063553
Running ISTA on the L1 reg. problem...  Elapsed time is 18.170983 seconds.
Running FISTA on the L1 reg. problem... Elapsed time is 0.222009 seconds.
```

- $\lambda = 10^{-5}$

```
Solving the L1 regularized problem (CVX)... ||x_star1 - x_s|| = 0.000005
Solving the L2 regularized problem (CVX)... ||x_star2 - x_s|| = 0.034183
Running ISTA on the L1 reg. problem...  Elapsed time is 127.854344 seconds.
Running FISTA on the L1 reg. problem... Elapsed time is 0.803443 seconds.
```

So, the trade-off to choose the λ value is convergence rate vs accuracy.

7 Exercise 7

7.1 Questions a, b

The problems

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 \ , \\ \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & F(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \|\mathbf{D}\mathbf{x}\|_1 \end{aligned}$$

are solved via CVX where $\mathbf{D} \in \mathbb{R}^{(n-1)\times n}$ is given by

$$\mathbf{D} = \rho \begin{bmatrix} +1 & -1 & & & \\ & +1 & -1 & & \\ & & \ddots & \ddots & \\ & & & +1 & -1 \end{bmatrix}.$$

By solving those problems, we are trying to recover $\mathbf{x}_{\text{pwc}}$ which is a piece-wise constant vector.

The $\|\mathbf{D}\mathbf{x}\|_1$ penalty term promotes solutions where the vector $\mathbf{D}\mathbf{x}$ is sparse.

The vector $\mathbf{D}\mathbf{x}$ is a sparse vector iff the vector $\mathbf{x}$ is piece-wise constant. This means that the $L1$ regularized problem promotes solutions that are piece-wise constant.

In the case that there is no noise, if the problems are overdetermined (i.e. $m > n$) or if $m = n$, then it is expected that the solutions of both problems should be very accurate (in particular, the first one should be exact). If the problems are underdetermined (i.e. $m < n$), the it is

expected that the solution of the second problem to be a lot better than the one of the first problem.

Below are following some console outputs in the no noise case illustrating the above point. The regularization parameter ρ has the value 10^{-4}.

- $m = 320, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 0.000000
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.000085
```

- $m = 310, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 0.000000
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.000137
```

- $m = 300, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 0.000000
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.000778
```

- $m = 300, n = 310$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 2.510435
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.001400
```

- $m = 300, n = 320$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 4.254560
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.003380
```

Now, adding noise will affect the solutions of both problems but keeping the above behavior: in the overdetermined (or $m = n$) case it is expected that the two problems will have approximately the same solution, while in the underdetermined case it is expected that the second problem will have a better solution.

The "weak" and "strong" noise is relevant to the matrix $\mathbf{A}$, which, in this case, has standard deviation 1.

Here are some console outputs with weak noise (`noise_std` $= 0.05, \rho = 10^{-4}$):

- $m = 320, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 0.163712
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.162214
```

- $m = 310, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 0.303213
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.300060
```

- $m = 300, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 0.541919
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.450022
```

- $m = 300, n = 310$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 4.243350
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.368677
```

- $m = 300, n = 320$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 4.243350
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 0.368677
```

And here are some console outputs with strong noise (`noise_std` $= 0.5, \rho = 10^{-4}$):

- $m = 320, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 1.754108
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 1.752070
```

- $m = 310, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 3.451502
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 3.431416
```

- $m = 300, n = 300$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 13.082888
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 12.608170
```

- $m = 300, n = 310$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 3.130105
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 2.543071
```

- $m = 300, n = 320$

```
Solving the simple LS problem (CVX)...      ||x_star1 - x_pwc|| = 4.958195
Solving the regularized LS problem (CVX)... ||x_star2 - x_pwc|| = 1.764722
```

Hence, the underdetermined case is of interest.

7.2 Question c

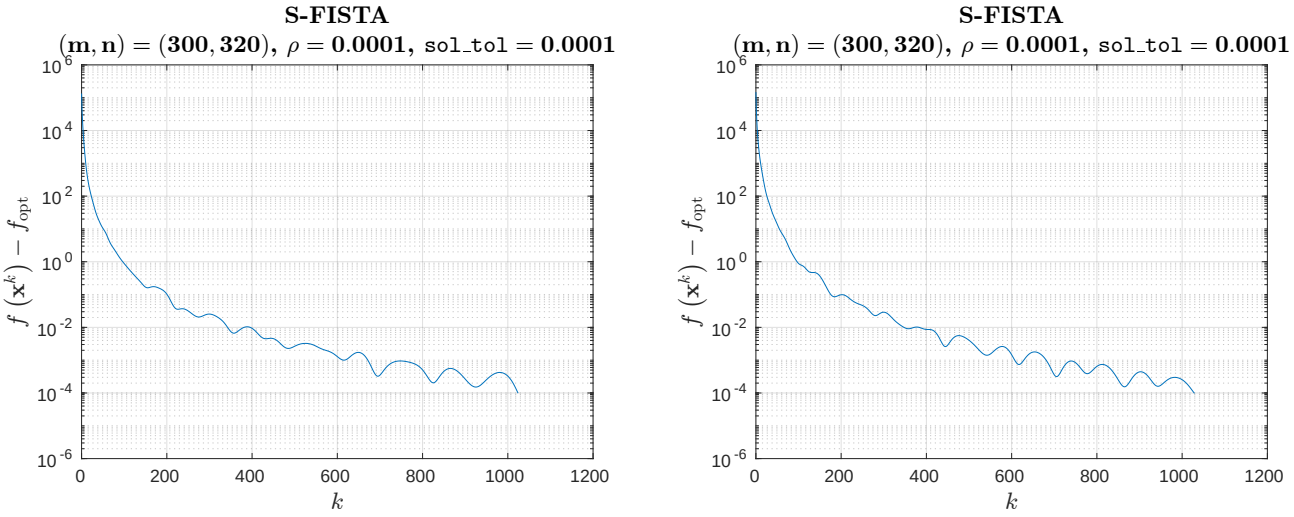

Figure 15: Noise standard deviation: 0.05 (left), 0.5 (right)

7.3 Question d

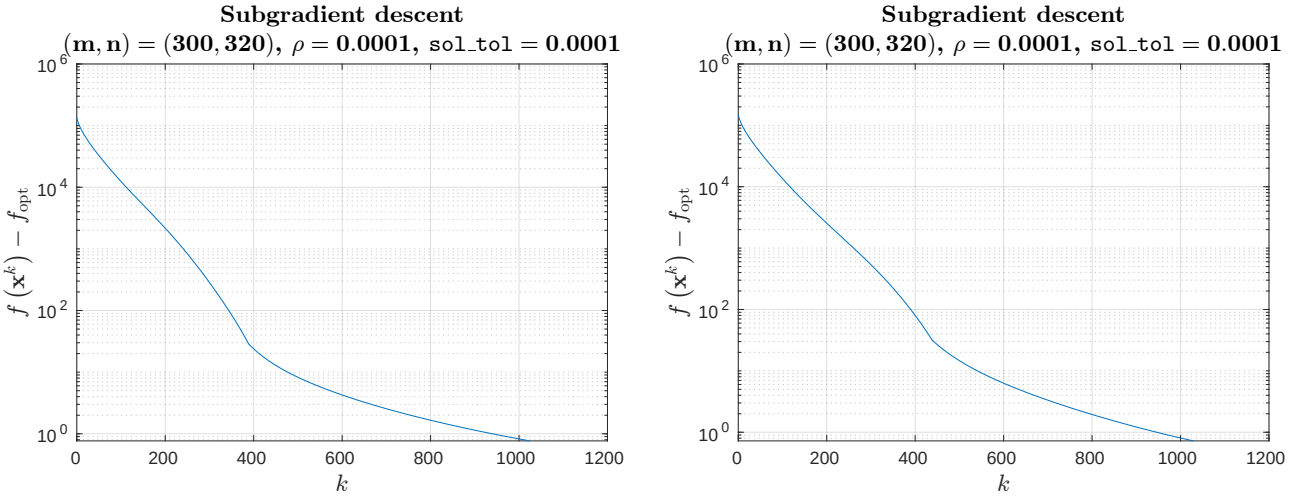

Figure 16: Noise standard deviation: 0.05 (left), 0.5 (right)

7.4 Question e

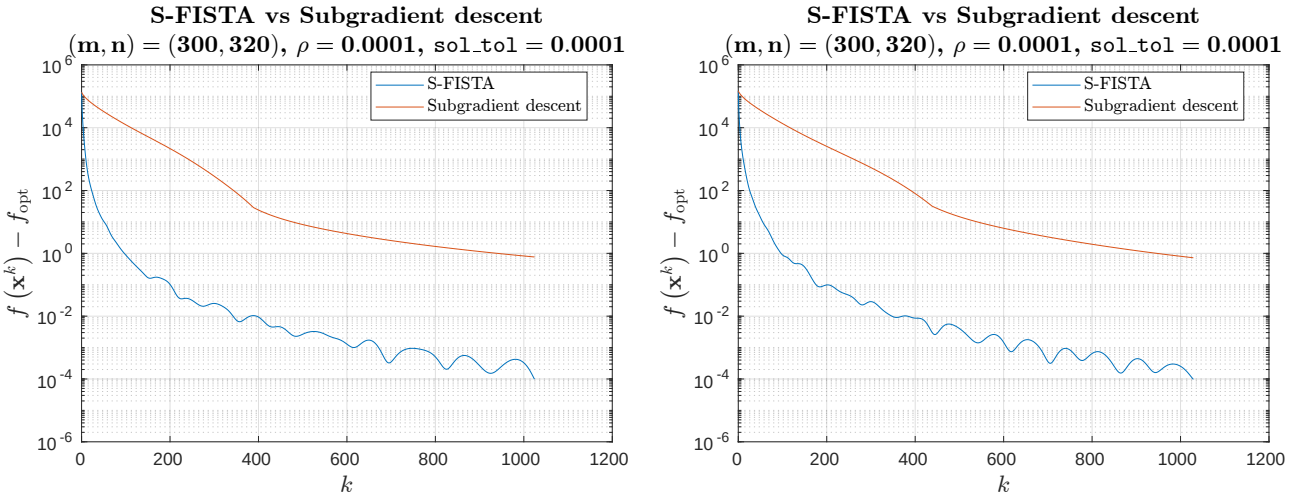

Figure 17: Noise standard deviation: 0.05 (left), 0.5 (right)

As is evident from the above convergence plots, S-FISTA converges faster than subgradient descent.

There is not a notable difference between the weak and strong noise cases, since the algorithms are trying to solve the optimization problem regardless of the level of noise present.

It has also been observed that S-FISTA requires a greater number of iterations to converge when $m \approx n$. Therefore, in such cases, it may be advisable to slightly relax the solution tolerance.

The convergence rate of S-FISTA depends on the ρ parameter, too. If ρ is too small (e.g. $\rho = 10^{-7}$) or too big (e.g. $\rho = 0.1$), then S-FISTA is becoming slow.

8 Exercise 8

8.1 Questions a, b

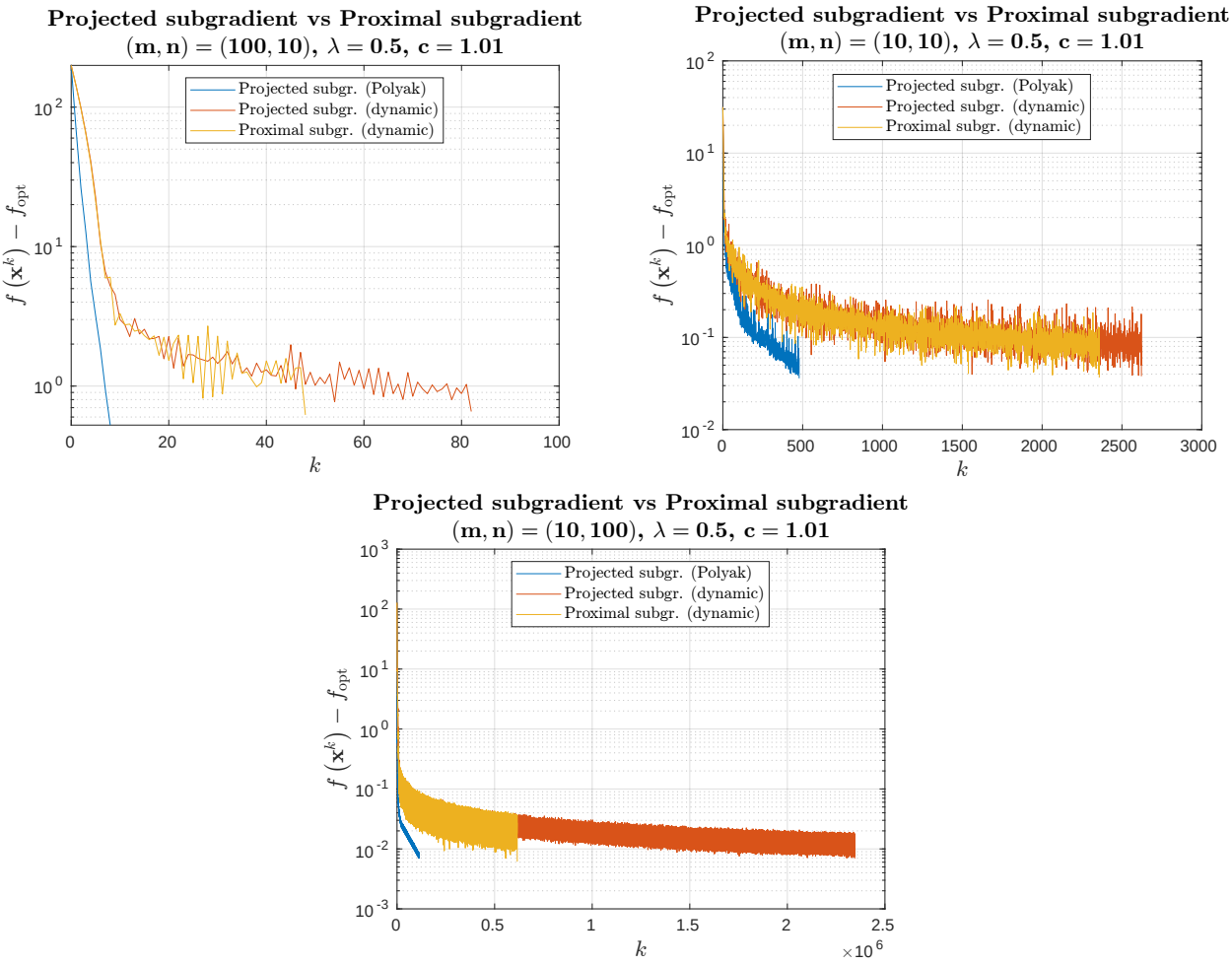

Figure 18

Convergence is achieved quickly in every case when using the Polyak stepsize, as it is the optimal stepsize. However, its computation requires knowledge of the optimal value of the problem.

Regarding the dynamic stepsize, it is observed that both algorithms perform quickly when $m \gg n$, since the problem is overdetermined in this case. For the scenario where $m \approx n$ the algorithms seem to converge in a similar way. However, in the case where $m \ll n$, the proximal subgradient algorithm tends to converge faster than the projected subgradient algorithm.

8.2 Question c

Goal: smooth $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1$ where $\mathbf{A} \in \mathbb{R}^{m \times n},\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{b} \in \mathbb{R}^m$.

Let $h : \mathbb{R}^m \mapsto \mathbb{R},\ h(\mathbf{y}) = \|\mathbf{y}\|_1$.

From Example 10.54 we have that

$$h_\mu(\mathbf{y}) = M_h^\mu(\mathbf{y}) = \sum_{i=1}^{m} H_\mu(y_i)$$

is $\frac{1}{\mu}$-smooth approximation of h with parameters $\left(1, \frac{m}{2}\right)$ where M_h^μ is the Moreau envelope of h and

$$H_\mu(\mathbf{x}) = \begin{cases} \dfrac{1}{2\mu}\|\mathbf{x}\|_2^2 & \text{if } \|\mathbf{x}\|_2 \leq \mu \\ \|\mathbf{x}\|_2 - \dfrac{\mu}{2} & \text{if } \|\mathbf{x}\|_2 > \mu \end{cases}$$

is the Huber function.

Let $\mathbf{y} = \mathbf{A}\mathbf{x} - \mathbf{b}$, so $h(\mathbf{A}\mathbf{x} - \mathbf{b}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 = f(\mathbf{x})$.

Also, let $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix}$.

From Theorem 10.46 (b) we have that

$$\begin{aligned} f_\mu(\mathbf{x}) &= h_\mu(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &= \sum_{i=1}^{m} H_\mu((\mathbf{A}\mathbf{x} - \mathbf{b})_i) \\ &= \sum_{i=1}^{m} H_\mu\left(\mathbf{a}_i^T\mathbf{x} - b_i\right) \end{aligned}$$

is $\frac{1}{\mu}$-smooth approximation of f with parameters $\left(\|\mathbf{A}\|_{2,2}^2, \frac{m}{2}\right)$.

In order to find the $\nabla f_\mu(\mathbf{x})$, we are going to need $\frac{dH_\mu}{dx}$:

$$\begin{aligned}
\frac{d}{dx}(H_\mu(x)) &= \begin{cases} \frac{1}{\mu}x & \text{if } |x| \le \mu \\ \operatorname{sgn}(x) & \text{if } |x| > \mu \end{cases} \\
&= \begin{cases} \frac{1}{\mu}\operatorname{sgn}(x)|x| & \text{if } |x| \le \mu \\ \frac{1}{\mu}\operatorname{sgn}(x)\mu & \text{if } |x| > \mu \end{cases} \\
&= \frac{1}{\mu}\operatorname{sgn}(x) \cdot \begin{cases} |x| & \text{if } |x| \le \mu \\ \mu & \text{if } |x| > \mu \end{cases} \\
&= \frac{1}{\mu}\operatorname{sgn}(x) \cdot \min\{|x|, \mu\} \\
&= \frac{1}{\mu}\operatorname{sgn}(x)(|x| + \min\{0, \mu - |x|\}) \\
&= \frac{1}{\mu}\operatorname{sgn}(x)(|x| - \max\{0, |x| - \mu\}) \\
&= \frac{1}{\mu}(\underbrace{\operatorname{sgn}(x)|x|}_{x} - \underbrace{\operatorname{sgn}(x)\max\{0, |x| - \mu\}}_{\mathcal{T}_\mu(x)}) \\
&= \frac{1}{\mu}(x - \mathcal{T}_\mu(x)).
\end{aligned}$$

So, the k-th element of $\nabla f_\mu(\mathbf{x})$ is

$$\begin{aligned}
(\nabla f_\mu(\mathbf{x}))_k = \frac{\partial}{\partial x_k}(f_\mu(\mathbf{x})) &= \sum_{i=1}^{m} \frac{\partial}{\partial x_k}\Big(H_\mu\Big(\mathbf{a}_i^T\mathbf{x} - b_i\Big)\Big) \\
&= \sum_{i=1}^{m} \frac{d}{d(\mathbf{a}_i^T\mathbf{x} - b_i)}\Big(H_\mu\Big(\mathbf{a}_i^T\mathbf{x} - b_i\Big)\Big) \cdot \underbrace{\frac{\partial}{\partial x_k}\Big(\mathbf{a}_i^T\mathbf{x} - b_i\Big)}_{a_{i,k}} \\
&= \sum_{i=1}^{m} a_{i,k} \cdot \frac{1}{\mu}\Big(\Big(\mathbf{a}_i^T\mathbf{x} - b_i\Big) - \mathcal{T}_\mu\Big(\mathbf{a}_i^T\mathbf{x} - b_i\Big)\Big)
\end{aligned}$$

Let $y_i = \mathbf{a}_i^T\mathbf{x} - b_i \iff \mathbf{y} = \mathbf{A}\mathbf{x} - \mathbf{b}$, so

$$\begin{aligned}
\nabla f_\mu(\mathbf{x}) &= \sum_{i=1}^{m} \mathbf{a}_i \cdot \frac{1}{\mu}(y_i - \mathcal{T}_\mu(y_i)) \\
&= \frac{1}{\mu}\mathbf{A}^T(\mathbf{y} - \mathcal{T}_\mu(\mathbf{y})) \\
&= \frac{1}{\mu}\mathbf{A}^T((\mathbf{A}\mathbf{x} - \mathbf{b}) - \mathcal{T}_\mu(\mathbf{A}\mathbf{x} - \mathbf{b})).
\end{aligned}$$

8.3 Question d

Goal: smooth $g(\mathbf{x}) = \|\mathbf{x}\|_1$ where $\mathbf{x} \in \mathbb{R}^n$.

From Example 10.54 we have that

$$g_\mu(\mathbf{x}) = M_g^\mu(\mathbf{x}) = \sum_{i=1}^{n} H_\mu(x_i)$$

is $\frac{1}{\mu}$-smooth approximation of g with parameters $\left(1, \frac{n}{2}\right)$, and, from question c, we get that

$$\nabla g_\mu(\mathbf{x}) = \frac{1}{\mu}(\mathbf{x} - \mathcal{T}_\mu(\mathbf{x})).$$

Now, we want to smooth F. Making use of Theorem 10.46 (a) we have that

$$F_\mu(\mathbf{x}) = f_\mu(\mathbf{x}) + \lambda g_\mu(\mathbf{x})$$

is $\frac{1}{\mu}$-smooth approximation of F with parameters $\left(\|\mathbf{A}\|_{2,2}^2 + \lambda, \frac{m}{2} + \lambda\frac{n}{2}\right)$.

The corresponding convergence plots for S-FISTA follow:

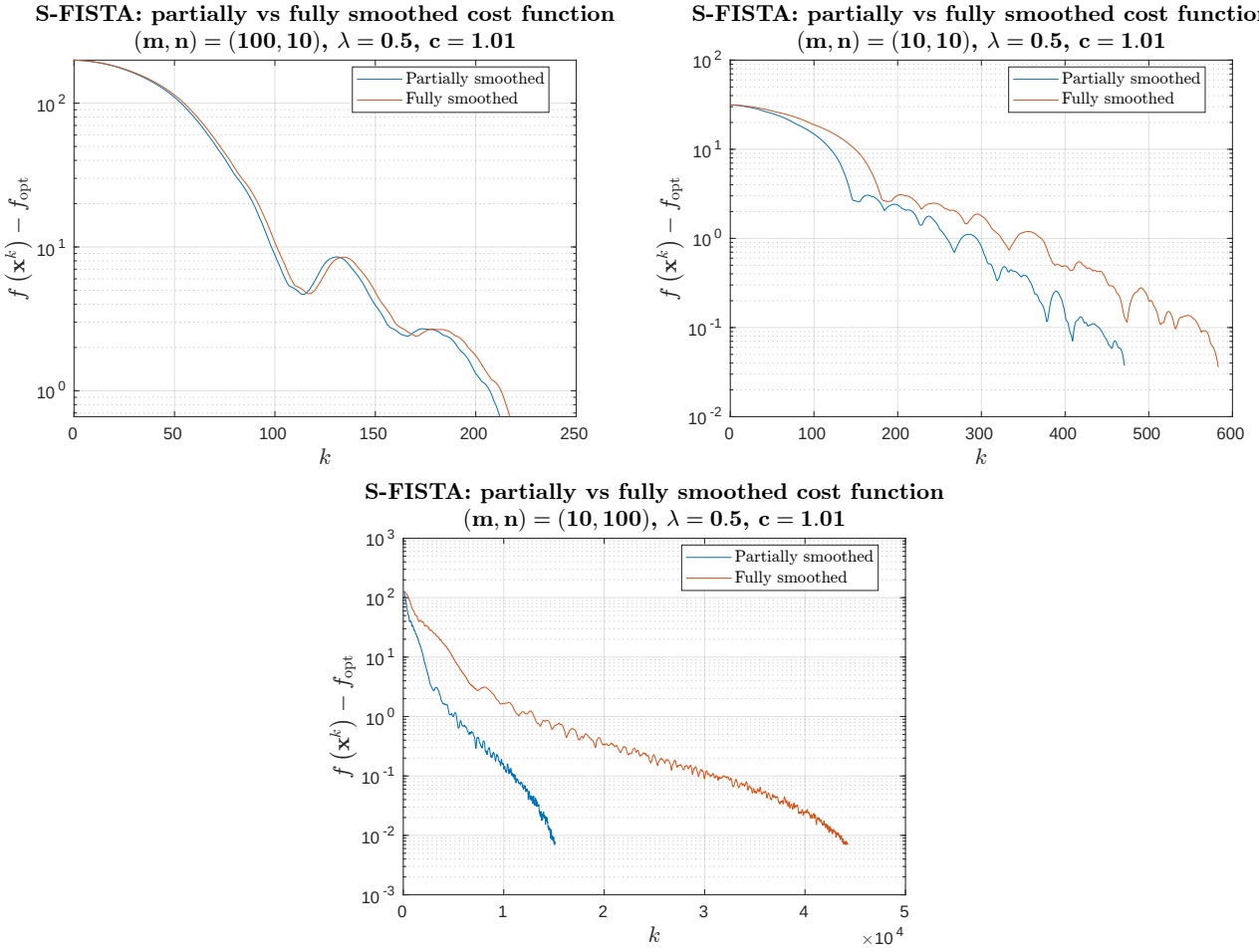

Figure 19

In the case where $m \gg n$ (overdetermined), S-FISTA on partially and fully smoothed cost function seem to have similar convergence rate.

But in any case, smoothing just the first part of the cost function should be preferred over smoothing the whole function.

And here are the convergence plots for all algorithms together:

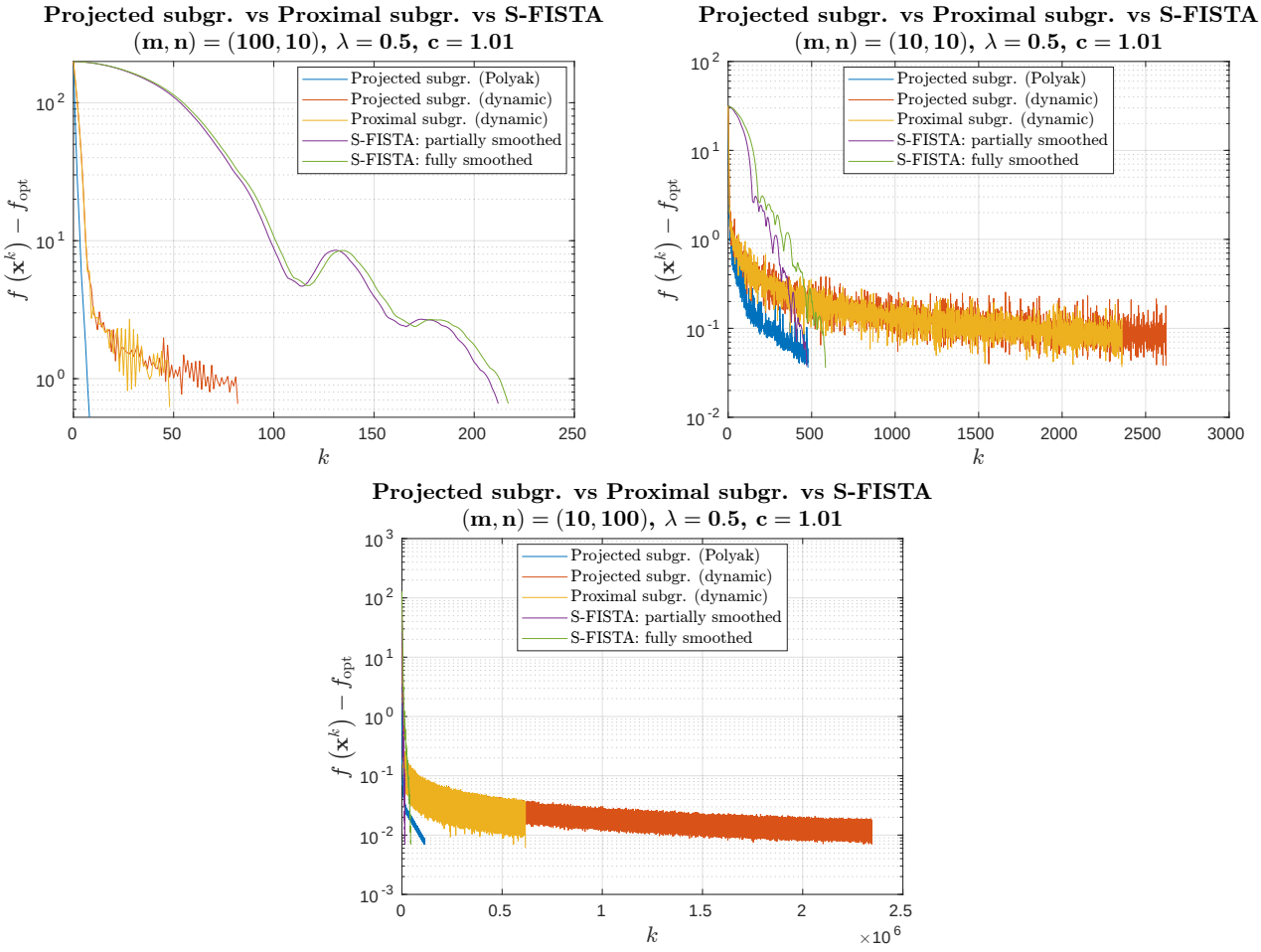

Figure 20

In the first plot ($m \gg n$), projected and proximal subgradient algorithms have converged earlier than S-FISTA due to their rapid convergence at their first iterations. However, if greater accuracy is required, the performance of projected and proximal subgradient algorithms changes after the first few iterations and S-FISTA converge at a much larger rate than them, as it happens at the rest plots.

In every case, S-FISTA with partially smoothed cost function is a better choice for both accuracy and convergence rate.

The parameter λ also has influence on the convergence rate of all the algorithms.

9 Exercise 9

9.1 Question a

The problem

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

is solved via CVX.

9.2 Question b

From question 8d we have that

$$h_\mu(\mathbf{x}) = M_h^\mu(\mathbf{x}) = \sum_{i=1}^{n} H_\mu(x_i)$$

is $\frac{1}{\mu}$-smooth approximation of h with parameters $\left(1, \frac{n}{2}\right)$, and

$$\nabla h_\mu(\mathbf{x}) = \frac{1}{\mu}(\mathbf{x} - \mathcal{T}_\mu(\mathbf{x})).$$

9.3 Question c

The convergence plots of both the S-FISTA method (from the previous question) and the subgradient descent method (from this question) are presented and compared in the following question.

9.4 Question d

In this problem, it is required that $m \leq n$. This is because the matrix $\mathbf{A}$ must have full row rank to ensure the constraint $\mathbf{A}\mathbf{x} = \mathbf{b}$ is feasible.

Therefore, experiments have been conducted with $m \ll n$ and $m \lessapprox n$.

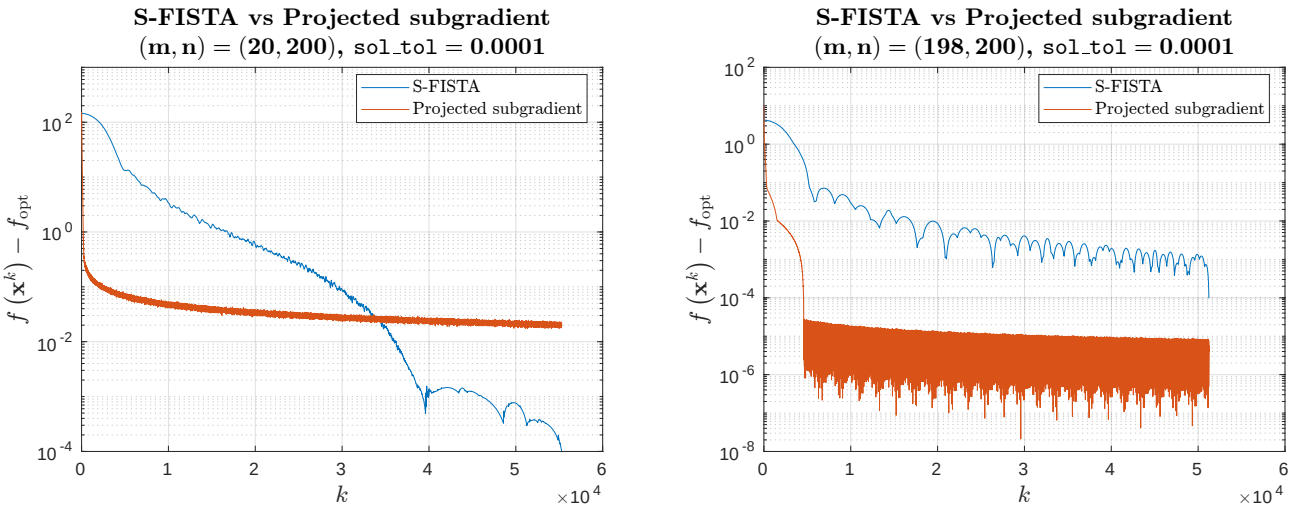

Figure 21

Here, it is observed that S-FISTA (FISTA with no proximal step on the smoothed cost function) and projected subgradient have different convergence behaviors, depending on the case.

In the scenario where $m \ll n$, the projected subgradient method initially converges faster than S-FISTA during the first few iterations. However, as the process continues, S-FISTA catches up and eventually converges faster due to the steeper trajectory of its convergence plot. So, for low solution tolerance, S-FISTA should be prefered in this case.

Conversely, in the case where $m \lessapprox n$, the projected subgradient method consistently outperforms S-FISTA in terms of convergence rate throughout the entire process. Hence, the projected subgradient descent method should be prefered in this case.

10 Exercise 10

10.1 Question a

The problem

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 \\ \text{subject to} \quad & \mathbf{x} \in \Delta_n = \{\mathbf{r} \in \mathbb{R}^n : \mathbf{r} \geq \mathbf{0},\ \textstyle\sum_{i=1}^n r_i = 1\} \end{aligned}$$

is solved via CVX.

10.2 Question b, c

From Example 9.19, for the mirror descent algorithm we have that

$$t_k = \frac{\sqrt{2}}{\|f'(\mathbf{x}^k)\|_\infty \sqrt{k+1}} \qquad \text{(dynamic stepsize)}$$

and

$$x_i^{k+1} = \frac{x_i^k e^{-t_k f_i'(\mathbf{x}^k)}}{\sum_{j=1}^n x_j^k e^{-t_k f_j'(\mathbf{x}^k)}}, \qquad i = 1, \ldots, n,$$

where $f_i'\left(\mathbf{x}^k\right)$ is the i-th element of $f'\left(\mathbf{x}^k\right)$.

So, the vector $\mathbf{x}^{k+1}$ will be

$$\mathbf{x}^{k+1} = \frac{1}{\exp(-t_k f'(\mathbf{x}^k))^T \mathbf{x}^k} \left(\mathbf{x}^k \odot \exp\left(-t_k f'\left(\mathbf{x}^k\right)\right)\right).$$

Note: In the Example 9.19 it is mentioned that $\mathbf{A} \in \mathbb{R}^{n \times n}$, but in this exercise $\mathbf{A} \in \mathbb{R}^{m \times n}$.

10.3 Question d, e

Experiments have been conducted with $m \ll n$, $m \lessapprox n$, $m \gg n$. The results are following:

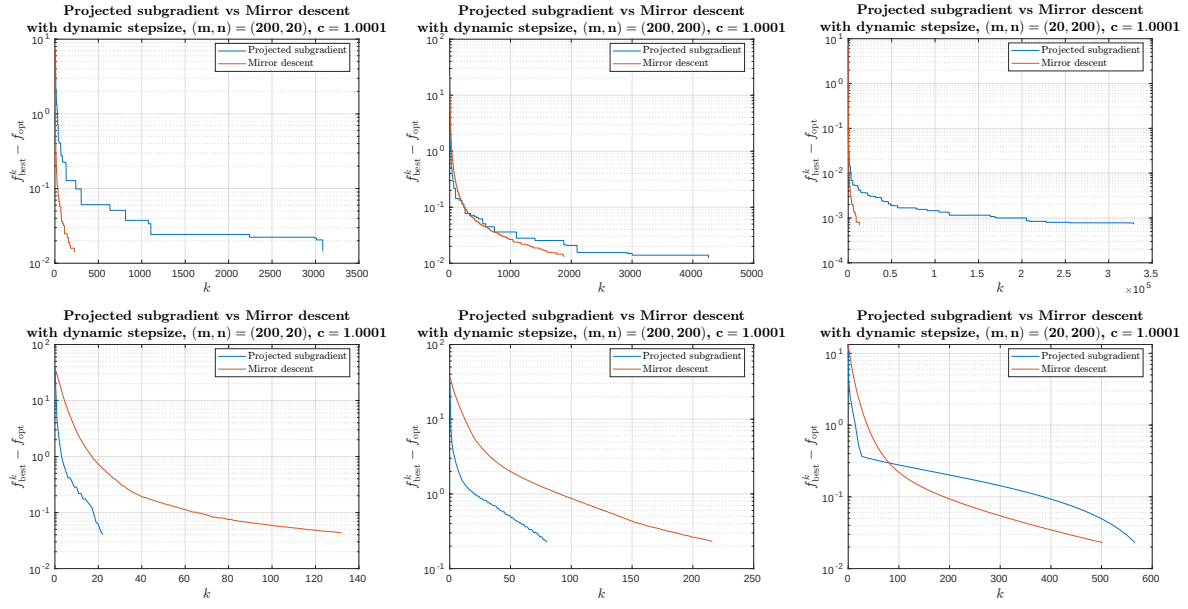

Figure 22: In the first row of figures, $\mathbf{b}$ was generated randomly; in the second row, $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}} + \mathbf{e}$, where $\mathbf{e}$ is noise.

In the scenario where $\mathbf{b}$ was generated randomly, it appears that the mirror descent algorithm converges faster.

In the other case, where $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{true}} + \mathbf{e}$, the projected subgradient algorithm seems to converge faster, except in the $m \ll n$ case where the two algorithms yield similar results.

11 Exercise 11

11.1 Question a

The problem to solve

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

can be expressed equivalently as

$$\begin{aligned} \underset{\mathbf{x}\in\text{dom}(f)\,,\,\mathbf{z}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x} = \mathbf{z}\ , \end{aligned} \tag{11.1}$$

where $g(\mathbf{z}) = \delta_{\mathbb{R}^n_+}(\mathbf{z})$ and $\text{dom}(f) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$.

The augmented Lagrangian for (11.1) is

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{x} - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z}\|_2^2,$$

and after performing the substitution $\mathbf{u} = \frac{1}{\rho}\mathbf{y}$ (and after some algebra), we get the scaled augmented Lagrangian

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{z} + \mathbf{u}\|_2^2 - \frac{\rho}{2}\|\mathbf{u}\|_2^2$$

which is a more preferable form for this problem; this will become obvious from the ADMM

updates that follow:

$$\begin{aligned}\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} L_\rho\Big(\mathbf{x}, \mathbf{z}^k, \mathbf{u}^k\Big) \\ &= \arg\min_{\mathbf{x}\in\mathbb{R}^n \,|\, \mathbf{A}\mathbf{x}=\mathbf{b}} \left\{ f(\mathbf{x}) + g\Big(\mathbf{z}^k\Big) + \frac{\rho}{2}\Big\|\mathbf{x} - \mathbf{z}^k + \mathbf{u}^k\Big\|_2^2 - \frac{\rho}{2}\Big\|\mathbf{u}^k\Big\|_2^2 \right\} \\ &= \arg\min_{\mathbf{x}\in\mathbb{R}^n \,|\, \mathbf{A}\mathbf{x}=\mathbf{b}} \left\{ \mathbf{c}^T\mathbf{x} + \frac{\rho}{2}\Big\|\mathbf{x} - \mathbf{z}^k + \mathbf{u}^k\Big\|_2^2 \right\} \end{aligned} \tag{11.2}$$

$$\begin{aligned}\mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}} L_\rho\Big(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{u}^k\Big) \\ &= \arg\min_{\mathbf{z}\in\mathbb{R}^n} \left\{ f\Big(\mathbf{x}^{k+1}\Big) + g(\mathbf{z}) + \frac{\rho}{2}\Big\|\mathbf{x}^{k+1} - \mathbf{z} + \mathbf{u}^k\Big\|_2^2 - \frac{\rho}{2}\Big\|\mathbf{u}^k\Big\|_2^2 \right\} \\ &= \arg\min_{\mathbf{z}\in\mathbb{R}^n} \left\{ \delta_{\mathbb{R}^n_+}(\mathbf{z}) + \frac{\rho}{2}\Big\|\mathbf{x}^{k+1} - \mathbf{z} + \mathbf{u}^k\Big\|_2^2 \right\} \\ &= \arg\min_{\mathbf{z}\in\mathbb{R}^n_+} \left\{ \Big\|\mathbf{x}^{k+1} - \mathbf{z} + \mathbf{u}^k\Big\|_2^2 \right\} \end{aligned} \tag{11.3}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{x}^{k+1} - \mathbf{z}^{k+1} \tag{11.4}$$

11.2 Question b

The optimization subproblem (11.2) is a quadratic problem with affine equality constraints. The gradient of the cost function is

$$\nabla_{\mathbf{x}}\left(\mathbf{c}^T\mathbf{x} + \frac{\rho}{2}\Big\|\mathbf{x} - \mathbf{z}^k + \mathbf{u}^k\Big\|_2^2\right) = \mathbf{c} + \rho\Big(\mathbf{x} - \mathbf{z}^k + \mathbf{u}^k\Big),$$

hence, the KKT conditions are

$$\begin{aligned}\mathbf{c} + \rho\Big(\mathbf{x}^* - \mathbf{z}^k + \mathbf{u}^k\Big) + \mathbf{A}^T\mathbf{v}^* = \mathbf{0} \iff \rho\mathbf{x}^* + \mathbf{A}^T\mathbf{v}^* &= \rho\mathbf{z}^k - \rho\mathbf{u}^k - \mathbf{c} \\ \mathbf{A}\mathbf{x}^* &= \mathbf{b}\end{aligned}$$

or, equivalently,

$$\begin{bmatrix}\rho\mathbf{I}_n & \mathbf{A}^T \\ \mathbf{A} & \mathbf{O}\end{bmatrix}\begin{bmatrix}\mathbf{x}^* \\ \mathbf{v}^*\end{bmatrix} = \begin{bmatrix}\rho\mathbf{z}^k - \rho\mathbf{u}^k - \mathbf{c} \\ \mathbf{b}\end{bmatrix}.$$

Since $\mathbf{I}_n \succ 0$, if $\mathbf{A}$ is full row rank, i.e. $\operatorname{rank}(\mathbf{A}) = m$, then (it can be proven that) the above matrix is invertible. Using the matrix inversion lemma for block matrices we get

$$\begin{bmatrix}\rho\mathbf{I}_n & \mathbf{A}^T \\ \mathbf{A} & \mathbf{O}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{I}_n - \mathbf{A}^T\Big(\mathbf{A}\mathbf{A}^T\Big)^{-1}\mathbf{A} & \mathbf{A}^T\Big(\mathbf{A}\mathbf{A}^T\Big)^{-1} \\ \Big(\mathbf{A}\mathbf{A}^T\Big)^{-1}\mathbf{A} & -\Big(\mathbf{A}\mathbf{A}^T\Big)^{-1}\end{bmatrix},$$

so, finally, the result is

$$\begin{aligned}
\mathbf{x}^* = \left(\begin{bmatrix}\mathbf{x}^*\\ \mathbf{v}^*\end{bmatrix}\right)_{1:n} &= \left(\begin{bmatrix}\mathbf{I}_n - \mathbf{A}^T\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\mathbf{A} & \mathbf{A}^T\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\\ \big(\mathbf{A}\mathbf{A}^T\big)^{-1}\mathbf{A} & -\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\end{bmatrix}\begin{bmatrix}\rho\mathbf{z}^k - \rho\mathbf{u}^k - \mathbf{c}\\ \mathbf{b}\end{bmatrix}\right)_{1:n}\\
&= \left(\begin{bmatrix}\mathbf{I}_n - \mathbf{A}^T\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\mathbf{A} & \mathbf{A}^T\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\\ \big(\mathbf{A}\mathbf{A}^T\big)^{-1}\mathbf{A} & -\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\end{bmatrix}\right)_{1:n,:}\begin{bmatrix}\rho\mathbf{z}^k - \rho\mathbf{u}^k - \mathbf{c}\\ \mathbf{b}\end{bmatrix}\\
&= \begin{bmatrix}\mathbf{I}_n - \mathbf{A}^T\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\mathbf{A} & \mathbf{A}^T\big(\mathbf{A}\mathbf{A}^T\big)^{-1}\end{bmatrix}\begin{bmatrix}\rho\mathbf{z}^k - \rho\mathbf{u}^k - \mathbf{c}\\ \mathbf{b}\end{bmatrix}.
\end{aligned}$$

The optimization subproblem (11.3) expressed in the known form is

$$\begin{aligned}
&\underset{\mathbf{z}\in\mathbb{R}^n}{\text{minimize}} && \tfrac{1}{2}\left\|\mathbf{z} - \mathbf{w}^k\right\|_2\\
&\text{subject to} && -z_i \le 0 \quad \forall\, i \in \{1,\dots,n\}\ ,
\end{aligned}$$

where $\mathbf{w}^k = \mathbf{x}^{k+1} + \mathbf{u}^k$.

The gradients of the cost function and the constraint functions are, respectively,

$$\nabla_{\mathbf{z}}\left(\frac{1}{2}\left\|\mathbf{z} - \mathbf{w}^k\right\|_2\right) = \mathbf{z} - \mathbf{w}^k\ , \qquad \nabla_{\mathbf{z}}(-z_i) = \nabla_{\mathbf{z}}\big(-\mathbf{e}_i^T\mathbf{z}\big) = -\mathbf{e}_i$$

Hence, the KKT conditions of this problem are

$$\mathbf{z}^* - \mathbf{w}^k + \sum_{i=1}^{n}\lambda_i^*(-\mathbf{e}_i) = 0 \iff \mathbf{z}^* = \mathbf{w}^k + \boldsymbol{\lambda}^* \tag{11.5}$$

$$\boldsymbol{\lambda}^* \ge \mathbf{0} \tag{11.6}$$

$$\lambda_i^* z_i^* = 0 \qquad \forall\, i \in \{1,\dots,n\} \tag{11.7}$$

$$\mathbf{z} \ge \mathbf{0}. \tag{11.8}$$

For $i \in \{1,\dots,n\}$:

- If $w_i^k < 0$, from (11.5) we have $z_i^* < \lambda_i^*$ which means that $\lambda_i^* > 0$ (overwise, $z_i^* < 0$ which contradicts (11.8)).

  So, from (11.7) we have that $z_i^* = 0$.

- If $w_i^k > 0$, from (11.5) we have $z_i^* > \lambda_i^*$ which means that $z_i^* > 0$ (overwise, $\lambda_i^* < 0$ which contradicts (11.6)).

  So, from (11.7) we have that $\lambda_i^* = 0$, so from (11.5) we have that $z_i^* = w_i^k$.

- If $w_i^k = 0$, from (11.5) we have $z_i^* = \lambda_i^*$, so from (11.7) we have that $z_i^* = 0$.

In short,

$$z_i^* = \max\left\{0, w_i^k\right\}$$

or

$$\begin{aligned}\mathbf{z}^* &= P_{\mathbb{R}^n_+}\left(\mathbf{w}^k\right) \\ &= P_{\mathbb{R}^n_+}\left(\mathbf{x}^{k+1} + \mathbf{u}^k\right).\end{aligned}$$

11.3 Question c

The implementation of ADMM is in the MATLAB file `admm_alg.m`.

Note: The performance of the algorithm is highly sensitive to the parameter ρ. Even a slight modification to its value can significantly impact the algorithm's speed, even in lower dimensions. For instance, when $m = 30$, $n = 300$, `sol_tol` $= 10^{-3}$ and $\rho = 1.1$, the algorithm gets stuck.

11.4 Question d

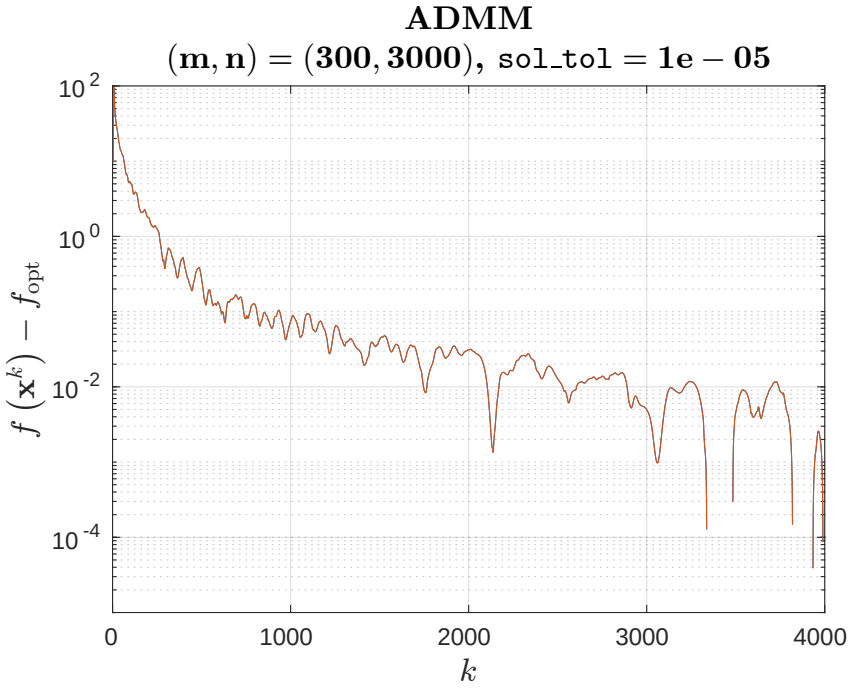

Figure 23: ADMM convergence plot: there were negative values that where ignored.

12 Exercise 12

The solution is in the MATLAB file `Ex12.m`.

13 Exercise 13

13.1 Question a

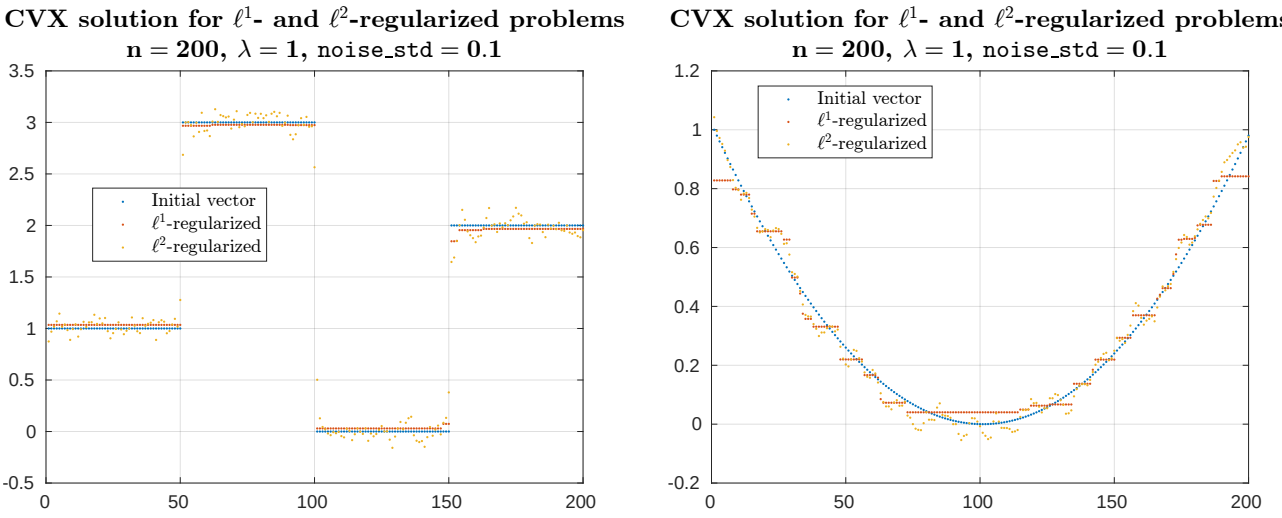

Figure 24: Left: piece-wise constant vector case. Right: smooth vector case.

The explanation is the same as in question 7b: the problem with the $\|\mathbf{Dx}\|_1$ penalty term promotes solutions that are piece-wise constant.

This explains why in the case of a piece-wise constant vector, the $L1$ regularized problem provides a superior solution. Similarly, in the smooth vector scenario, it is the reason why the "step" patterns are observed in the solution to the $L1$ regularized problem.

13.2 Question b

In order to avoid all matrix-vector multiplications with $\mathbf{D}$, we have that $\mathbf{D} \in \mathbb{R}^{(n-1)\times n}$ with

$$\mathbf{D} = \begin{bmatrix} +1 & -1 & & & \\ & +1 & -1 & & \\ & & \ddots & \ddots & \\ & & & +1 & -1 \end{bmatrix} \qquad \mathbf{D}^T = \begin{bmatrix} \mathbf{I}_{n-1} \\ \mathbf{0}_{n-1}^T \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n-1}^T \\ -\mathbf{I}_{n-1} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{I}_{n-1} & \mathbf{0}_{n-1} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n-1} & -\mathbf{I}_{n-1} \end{bmatrix}$$

so, we have that

$$\mathbf{Dx} = \begin{bmatrix} \mathbf{I}_{n-1} & \mathbf{0}_{n-1} \end{bmatrix} \begin{bmatrix} (\mathbf{x})_{1:n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{n-1} & -\mathbf{I}_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ (\mathbf{x})_{2:n} \end{bmatrix}$$
$$= (\mathbf{x})_{1:n-1} - (\mathbf{x})_{2:n}$$

and

$$\mathbf{D}^T\mathbf{y} = \begin{bmatrix}\mathbf{I}_{n-1}\\ \mathbf{0}_{n-1}^T\end{bmatrix}\mathbf{y} + \begin{bmatrix}\mathbf{0}_{n-1}^T\\ -\mathbf{I}_{n-1}\end{bmatrix}\mathbf{y}$$
$$= \begin{bmatrix}\mathbf{y}\\ 0\end{bmatrix} - \begin{bmatrix}0\\ \mathbf{y}\end{bmatrix}.$$

After running both DPG and FDPG for the $L1$ regularized problem (same number of iterations), the following results emerge:

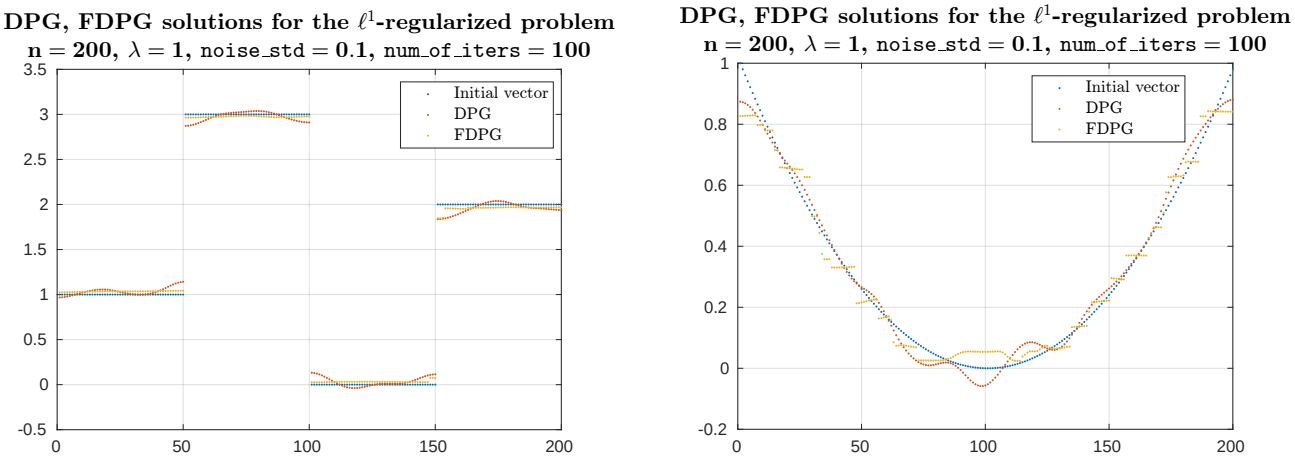

Figure 25: Left: piece-wise constant vector case. Right: smooth vector case.

Here are the convergence plots for both DPG and FDPG algorithms together:

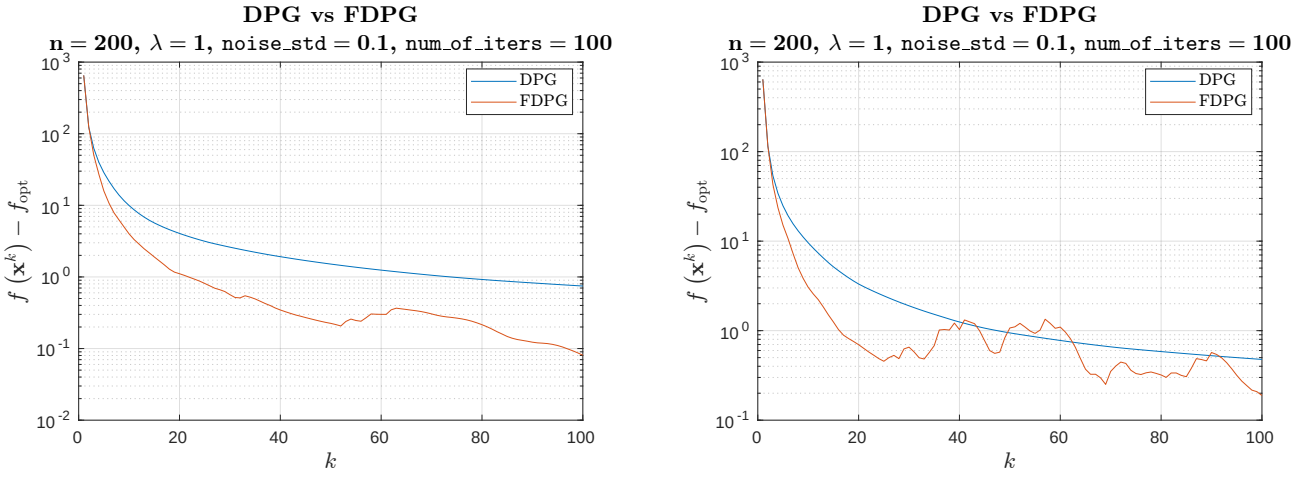

Figure 26: Left: piece-wise constant vector case. Right: smooth vector case.

In general, FDPG has higher convergence rate than DPG. However, it is evident that in case of the smooth vector, the results are evidently worse.