Sharif University of Technology
Department of Electrical Engineering
Convex Optimization I (25756-1)

CHW 2

Instructor: Dr. R. Amiri — Fall Semester 1403 — Max mark: 100

1. **FIR Filter Design (30 Pts)**

   Introduction to FIR Filters
   Finite Impulse Response (FIR) filters are a crucial component in digital signal processing. They are used to filter signals by allowing certain frequencies to pass while attenuating others. FIR filters are characterized by their finite duration impulse response, meaning that their output depends only on a finite number of input samples.

   Why Use Convex Optimization?
   The design of FIR filters involves determining the optimal filter coefficients to achieve a desired frequency response while minimizing certain performance criteria, such as the maximum deviation from the desired response or the energy of the coefficients. Convex optimization provides a powerful framework for solving these problems because:

   - **Convex Problems**: The optimization problems can often be formulated as convex problems, which have desirable properties such as guaranteed convergence to a global minimum.
   - **Efficiency**: Convex optimization techniques can efficiently handle large-scale problems and provide robust solutions.
   - **Flexibility**: It allows for the incorporation of various constraints, such as passband and stopband specifications.

   Mathematical Formulation The goal is to design an FIR filter with coefficients $h[n]$ for $n = 0, 1, \ldots, N$ that minimizes the maximum deviation from a desired frequency response. The desired frequency response $H_d(e^{j\omega})$ can be defined as:

   - **Passband**: For frequencies $|\omega| \leq \omega_p$, we want $|H(e^{j\omega})| \approx 1$
   - **Stopband**: For frequencies $|\omega| \geq \omega_s$, we want $|H(e^{j\omega})| \approx 0$

   Where:
   $$H(e^{j\omega}) = \sum_{n=0}^{N} h[n]e^{-jn\omega}$$

   The optimization problem can then be formulated as:
   $$\text{Minimize } \max_{\omega} |H(e^{j\omega}) - H_d(e^{j\omega})|$$
   subject to constraints on the passband and stopband deviations.

   Problem Statement Design a Low-Pass FIR Filter Using Convex Optimization.

   You are tasked with designing a low-pass FIR filter that meets specific frequency response requirements using convex optimization techniques in Python with CVXPY.

   Specifications:

1. Passband Frequency (f_p): 0.2 (normalized frequency)
2. Stopband Frequency (f_s): 0.3 (normalized frequency)
3. Passband Ripple: 0.05
4. Stopband Attenuation: 0.1
5. Filter Order: 20

Objective
Formulate this problem as a convex optimization problem where you minimize the maximum deviation from the desired frequency response while adhering to constraints on the filter coefficients.

Implementation Instructions:

1. Define the filter coefficients as optimization variables.
2. Set up the frequency response constraints based on the specifications.
3. Use CVXPY to solve the optimization problem.
4. Visualize the resulting filter's frequency response using Matplotlib.

Visualization Task: After implementing your code, experiment with varying:

- Passband frequency (f_p)
- Stopband frequency (f_s)
- Filter order (N)

Create plots for each variation and observe how these changes affect performance.

2. **Introduction to convex optimization algorithms, Part 1 (30 Pts)**
Note: Most of the text is devoted to introduce you to the basics of some optimization algorithms. If you are already familiar with them, skip these parts and go straight to the objective section.
Minimization algorithms are a crucial part of optimization problems. More importantly, using a suitable optimization algorithm results in significant reduction in computation time and memory, and are preferred over generic convex optimization solvers in many instances. Generic solvers need to handle even pathological problem formulations, which comes at the cost of efficiency.
For simplicity, first we will investigate two algorithms for the LASSO problem.

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \gamma\|x\|_1 \tag{1}$$

Proximal Gradient algorithm
In many cases, parts of the objective are not differentiable, in particular when we have indicator functions for constraints. In some other cases, the (sub)gradients exist but may be ill-behaved (a good example is the logarithm function). In these cases, we can break the function in two parts and proceed as follows:

$$\text{minimize} \quad f(\mathbf{x}) + g(\mathbf{x})$$

Where f is smooth and g is proper convex lower semi-continuous. Most usually, we want g to have a simple closed-form proximal operator:

$$\text{prox}_g = \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 + g(\mathbf{x}) \tag{2}$$

Now that we know the background, the proximal gradient algorithm is as follows:

$$\mathbf{x}^{k+1} := \operatorname{prox}_{\lambda^k g}\left(\mathbf{x}^k - \lambda^k \nabla f\left(\mathbf{x}^k\right)\right)$$

It converges with rate $O(1/k)$ when ∇f is Lipschitz continuous with constant L and step sizes are $\lambda^k = \lambda \in (0, 1/L]$

For the LASSO problem, we have calculated the steps for you:

$$\mathbf{x}^{k+1} := \operatorname{prox}_{\lambda^k \gamma \|\cdot\|_1}\left(\mathbf{x}^k - \lambda^k \mathbf{A}^T\left(\mathbf{A}\mathbf{x}^k - \mathbf{b}\right)\right) \tag{3}$$

Where the proximal operator of $\|\cdot\|$ is simply the soft thresholding operator, as you can easily verify:

$$\operatorname{prox}_{\lambda\|\mathbf{y}\|} = \begin{cases} y_i - \lambda & y_i \geq \lambda \\ 0 & -\lambda \leq y_i \leq \lambda \\ y_i + \lambda & y_i \leq -\lambda \end{cases} \tag{4}$$

Next, we investigate another optimization algorithm called Alternating Direction Method of Multipliers (ADMM):

Consider again the problem

$$\text{minimize} \quad f(\mathbf{x}) + g(\mathbf{x})$$

We formulate the Augmented Lagrangian in the following way, and optimize it over each variable $\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}$ each time. The minimization steps can be written in the following equivalent form:

$$\begin{aligned} \mathcal{L}_\rho &= f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\mu}^T(\mathbf{x} - \mathbf{y}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 \\ \mathbf{x}^{k+1} &= \operatorname{prox}_{f/\rho}(\mathbf{y}^k - \boldsymbol{\mu}^k/\rho) \\ \mathbf{y}^{k+1} &= \operatorname{prox}_{g/\rho}(\mathbf{x}^{k+1} + \boldsymbol{\mu}^k/\rho) \\ \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + \rho(\mathbf{x}^{k+1} - \mathbf{y}^{k+1}) \end{aligned} \tag{5}$$

It should be noted that in many standard texts, a rescaled version exists using $\boldsymbol{\mu}/\rho$ in the expressions.

The steps for LASSO are:

$$\begin{aligned} \mathbf{x}^{k+1} &:= \left(I + \frac{1}{\rho}\mathbf{A}^T\mathbf{A}\right)^{-1}\left(\mathbf{y}^k - \boldsymbol{\mu}^k + \frac{1}{\rho}\mathbf{A}^T\mathbf{b}\right) = \left(\rho I + \mathbf{A}^T\mathbf{A}\right)^{-1}\left(\rho\mathbf{y}^k - \rho\boldsymbol{\mu}^k + \mathbf{A}^T\mathbf{b}\right) \\ \mathbf{y}^{k+1} &:= \operatorname{prox}_{\frac{\gamma}{\rho}\|\cdot\|_1}\left(\mathbf{x}^{k+1} + \boldsymbol{\mu}^k\right) \\ \boldsymbol{\mu}^{k+1} &:= \boldsymbol{\mu}^k + \mathbf{x}^{k+1} - \mathbf{y}^{k+1} \end{aligned} \tag{6}$$

Objectives: We aim to implement the three algorithms and compare their performance. Implementation should be in Python, using generic libraries as Numpy, Scipy, etc. Your implementations should be "generally efficient", that is, some basic and obvious steps toward efficiency should be taken into account. However, it is not a coding contest and simply avoiding common sense mistakes is enough.

1. Implement the loss function in CVXPY and solve the problem for different dimensions of the input matrix. Also report the time spent by CVXPY.
2. Implement the proximal gradient algorithm in Python, and repeat part 1. You are allowed to change the stepsize parameter as you deem fit, but include a few results for each stepsize.
3. Implement the ADMM algorithm for LASSO, and repeat part 1. Report the changes for a few different choices of ρ, for a fixed matrix dimension of your choice.

Introduction to convex optimization algorithms, Part 2 (50 Pts) [1]
Consider the problem of graph learning, in which we intend to infer a graph by having observations of signals over a graph. One popular approach is use the signal smoothness:

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_{i,j=1}^{N}(x_i - x_j)^2 w_{ij}$$

Where $\mathbf{L}$ and $\mathbf{W}$ are the Laplacian and weight matrix, respectively. We now investigate a popular cost function also implemented in GSPBOX MATLAB package:

$$\begin{array}{ll} \min_{\mathbf{W}\in\mathbb{R}^{s\times s}} & \|\mathbf{W}\circ\mathbf{Z}\|_{1,1} - \alpha\mathbf{1}^\top\log(\mathbf{W}\mathbf{1}) + \frac{\beta}{2}\|\mathbf{W}\|_F^2 \\ \text{s.t.} & W \geq \mathbf{0}, \mathbf{W} = \mathbf{W}^T, \operatorname{diag}(\mathbf{W}) = \mathbf{0} \end{array} \tag{1}$$

The above problem can be expressed only as a function the upper triangular elements of $\mathbf{W}$ only, in the following way:

$$\begin{array}{ll} \min_{\mathbf{w}\in\mathbb{R}^m} & 2\mathbf{z}^T w - \alpha\mathbf{1}^\top\log(\mathbf{Q}\mathbf{w}) + \beta\|\mathbf{w}\|_2^2 \\ \text{s.t.} & \mathbf{w} \geq 0 \end{array} \tag{7}$$

Where $\mathbf{Q}$ is a sparse binary matrix which satisfies $\mathbf{Q}\mathbf{w} = \mathbf{W}\mathbf{1}$. It has been proposed recently that we solve the problem via ADMM, that is, to add a constraint $\mathbf{Q}\mathbf{w} = \mathbf{v}$:

$$\begin{array}{ll} \min_{\mathbf{w}\in\mathbb{R}^m,\mathbf{v}\in\mathbb{R}^s} & f(\mathbf{w}) + g(\mathbf{v}) \\ \text{s.t.} & \mathbf{Q}\mathbf{w} = \mathbf{v} \end{array} \tag{8}$$

With $f(\mathbf{w}) = 2\mathbf{z}^T\mathbf{w} + \beta\|\mathbf{w}\|_2^2 + \mathbb{I}_{\{\mathbf{w}\geq 0\}}$ and $g(\mathbf{v}) = -\alpha\mathbf{1}^T\log(\mathbf{v})$.
Here, instead, for optimizing each ADMM subproblem, they use a *proximal gradient* step:

$$\begin{aligned} \mathbf{w}^{k+1} &= \operatorname{prox}_{\tau_1 f}\left(\mathbf{w}^k - \tau_1 t\mathbf{Q}^\top\left(\mathbf{Q}\mathbf{w}^k - \mathbf{v}^k - \frac{\boldsymbol{\lambda}^k}{t}\right)\right) = \max\left\{\tilde{\mathbf{w}}^{k+1}, \mathbf{0}\right\} \\ \tilde{w}^{k+1} &= \frac{w^k - \tau_1 tQ^\top\left(Qw^k - v^k - \frac{\boldsymbol{\lambda}^k}{t}\right) - 2\tau_1\mathbf{z}}{2\tau_1\beta + 1} \end{aligned} \tag{9}$$

$$\begin{aligned} \mathbf{v}^{k+1} &= \operatorname{prox}_{\tau_2 g}\left(\mathbf{v}^k + \tau_2 t\left(\mathbf{Q}\mathbf{w}^{k+1} - \mathbf{v}^k - \frac{\boldsymbol{\lambda}^k}{t}\right)\right) = \frac{\tilde{\mathbf{v}}^{k+1} + \sqrt{\left(\tilde{\mathbf{v}}^{k+1}\right)^2 + 4\alpha\tau_2\mathbf{1}}}{2} \\ \tilde{\mathbf{v}}^{k+1} &= (1 - \tau_2 t)\,\mathbf{v}^k + \tau_2 t\mathbf{Q}\mathbf{w}^{k+1} - \tau_2\boldsymbol{\lambda}^k \end{aligned} \tag{10}$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - t\left(\mathbf{Q}\mathbf{w}^{k+1} - \mathbf{v}^{k+1}\right) \tag{11}$$

Theoretical conditions on the stepsizes for convergence are: $\tau_1 < \frac{1}{t\sigma_{max}^2(\mathbf{Q})} = \frac{1}{2(N-1)t}$, $\tau_2 < \frac{1}{t}, t > 0$.
Use $r_p = \left\|t\mathbf{Q}^T\left(\mathbf{v}^{k+1} - \mathbf{v}^k\right)\right\|_2$ and $r_d = \left\|\mathbf{Q}\mathbf{w}^k - \mathbf{v}^k\right\|_2$ as convergence criteria.

Objective

1. Formulate the problem in CVXPY and evaluate its computation time for graphs with vertex numbers $N = 30, 100$.
2. Implement the ADMM algorithm and evaluate its performance for the same parameters as part 1. You are allowed to change τ_1, τ_2, t as you deem fit. Evaluate the time consumption of you algorithm the same way as part 1.
   Your algorithm should be implemented in Python, using generic libraries such as Numpy. No step should take time higher that $O(n^2)$. It is highly recommended to use vectorized Numpy functions for execution speed.

Note: You may find that in [1], some expressions are different from this file. This is due to the fact that there are some typos in the article, so be sure to use these equations.

References

[1] X. Wang, C. Yao, H. Lei and A. M. -C. So, ”An Efficient Alternating Direction Method for Graph Learning from Smooth Signals,” ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 2021, pp. 5380-5384, doi: 10.1109/ICASSP39728.2021.9414791. ,