Augmented Lagrangian and Alternating Direction Method of Multipliers : Theory, Algorithm, and Applications

Md. Toufiqur Rahman[1] & Rifat Bin Rashid[2]
[1]RUID: 236007519, [2]RUID: 237000174
Department of Electrical and Computer Engineering, Rutgers University
Email: {mr2103, rifat.rashid}@rutgers.edu

Abstract

This project investigates the mathematical foundations and algorithmic principles of the Augmented Lagrangian and Alternating Direction Method of Multipliers (ADMM) in the context of constrained optimization. Beginning with a generic constrained optimization formulation, the study systematically develops the theoretical framework by examining essential precursors of ADMM, including the Lagrangian function, Dual Ascent, Dual Decomposition, and the Augmented Lagrangian Method. The practical efficiency of these methods is demonstrated through three case studies: sparse regression via LASSO, image reconstruction, and a hybrid precoding design framework for millimeter wave MIMO systems.

All simulation files and result plots are available at GitHub repository.

1 Introduction

Optimization methods play a central role in modern signal processing and wireless communication systems. In particular, constrained optimization problems of the form:

$$\min_{x} \ f(x) \quad \text{s.t. } Ax = b, \ x \in \mathcal{C} \tag{1}$$

appear frequently in applications where f may be non-smooth or non-convex and $\mathcal{C}$ includes structural constraints. Solving such problems efficiently requires advanced methods beyond traditional Lagrangian duality.

This project focuses on the Augmented Lagrangian framework and the ADMM algorithm, which decomposes complex constrained problems into simpler subproblems. This decomposability, combined with ADMM's robustness, has established it as one of the most pivotal building blocks of distributed optimization [1]. By enabling coordinated problem-solving among distributed agents, ADMM ensures collective convergence to an optimal or near-optimal solution. While ADMM has strong theoretical guarantees under convexity [2], it also exhibits robustness in practice even when theoretical assumptions are relaxed [3], [4].

In machine learning, ADMM provides an efficient approach for sparse modeling techniques such as the LASSO (Least Absolute Shrinkage and Selection Operator), which performs feature selection while fitting linear models [5]. The method's decomposition structure allows it to handle high-dimensional datasets where traditional solvers may struggle, making it valuable for big data applications. By separating the optimization into parallelizable steps, ADMM enables scalable implementations that maintain statistical accuracy while improving computational efficiency.

As networks evolve toward massive MIMO architectures in 5G and beyond, base stations must solve challenging beamforming problems under strict latency constraints. ADMM addresses this by distributing the computational load across different processing units, allowing real-time optimization of transmission strategies while meeting power constraints and interference management requirements. The method's robustness to partial updates and its convergence properties make it particularly attractive for these dynamic wireless environments, often accompnained by deep neural network [6]. This report discuss a theoretical framework for application of ADMM for hybrid precoding design in MIMO system.

The rest of the report is organized as follows: The following section presents the theoretical background of ADMM through the precursors, Section 3 explains the framework for application of ADMM through three case studies along with simulation results. Finally, the study concludes at Section 4 with direction to possible future improvements.

2 Precursors

2.1 The Lagrangian

A constrained optimization problem in standard form:

$$\begin{array}{rll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \dots, m, \\ & h_i(x) = 0, & i = 1, \dots, p, \end{array} \tag{2}$$

where $x \in \mathbb{R}^n$ is the optimization variable. The domain of the problem is assumed to be nonempty:

$$\mathcal{D} = \bigcap_{i=0}^{m} \operatorname{dom} f_i \cap \bigcap_{i=1}^{p} \operatorname{dom} h_i,$$

and the optimal value is denoted by $p^\star$. Convexity of the problem is not required.

To incorporate the constraints into the objective, the *Lagrangian function* is defined as :

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x), \tag{3}$$

with domain $\operatorname{dom} L = \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$. Here, λ_i and ν_i are the Lagrange multipliers for the inequality and equality constraints, respectively. The vectors $\lambda \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^p$ are referred to as the dual variables. A crucial aspect of the Lagrangian is that it is linear (and thus both convex and concave) in the dual variables (λ, ν) for any fixed primal variable x.

The *Lagrangian* allows to combine the objective function and the constraints into a single scalar function. This unified formulation enables the use of techniques originally developed for unconstrained optimization (such as gradient-based methods) to be extended to problems with constraints. Moreover, the Lagrangian serves as the foundational tool for the following sections on dual ascent, dual decomposition, augmented Lagrangian methods, and ADMM. It is especially useful when the original constrained problem cannot be solved directly, for instance, due to the complexity of the equality constraints. In such cases, working with the dual function or the augmented Lagrangian often leads to more tractable or decomposable formulations as explained in following subsections.

2.2 Dual Ascent

For a convex optimization problem of this form:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b, \end{array} \tag{4}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $f : \mathbb{R}^n \to \mathbb{R}$ is convex,

the associated Lagrangian as defined in Eqn: 3 is

$$L(x, y) = f(x) + y^T(Ax - b), \tag{5}$$

and the dual function is given by

$$g(y) = \inf_x L(x, y) = -f^*(-A^T y) - b^T y, \tag{6}$$

where f^* is the convex conjugate of f. Since dual function is point-wise infimum of a concave function (Lagrangian as a linear function of dual variables exhibits both convex and concave nature simultaneously), it is always concave regardless of the nature of primal problem. The dual problem becomes:

$$\max_y \quad g(y),$$

with variable $y \in \mathbb{R}^m$. The dual function as well as the dual problem provides a lower bound on the primal value. When strong duality holds, the optimal value equals that of the primal. If $L(x, y^*)$ has a unique minimizer, the primal solution can be recovered via

$$x^* = \arg\min_x L(x, y^*).$$

The dual is solved by gradient ascent. Assuming g is differentiable, the gradient is computed by minimizing $L(x, y)$ with respect to x, then evaluating:

$$\nabla g(y) = Ax^+ - b.$$

This leads to the update rules:

$$x^{k+1} := \arg\min_x L(x, y^k), \tag{7}$$

$$y^{k+1} := y^k + \alpha^k(Ax^{k+1} - b), \tag{8}$$

where $\alpha^k > 0$ is the step size. Eqn. 7 minimizes the Lagrangian, while the Eqn. 8 updates the dual variable—interpreted as a price adjustment.

Even when g is not differentiable, this method still applies using subgradients. In such cases, the residual $Ax^{k+1} - b$ serves as a subgradient of $-g$, and the method is called the *dual subgradient method* [7].

2.2.1 Limitation of Dual Ascent

While Dual Ascent is a fundamental concept in optimization, a main limitation of this method is the need to solve $\arg\min_x L(x, y^k)$ at each iteration, which may be computationally intractable. If f is not strictly convex, the minimizer may be non-unique, and $\nabla g(y)$ may be undefined, making convergence unreliable in many practical settings.

2.3 Dual Decomposition

A key advantage of dual ascent is its potential for decentralization when the objective is separable. Consider the separable objective function:

$$f(x) = \sum_{i=1}^{N} f_i(x_i),$$

with $x = (x_1, \dots, x_N)$, where each $x_i \in \mathbb{R}^{n_i}$. Let the constraint matrix be partitioned as

$$A = [A_1 \ \cdots \ A_N], \quad \text{so that} \quad Ax = \sum_{i=1}^{N} A_i x_i.$$

Then the Lagrangian for Problem 4 becomes:

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} \left(f_i(x_i) + y^T A_i x_i - \frac{1}{N} y^T b \right), \tag{9}$$

which is separable across x_i. As a result, the x-minimization step can be split into independent subproblems, enabling parallel execution.

The dual decomposition algorithm proceeds as:

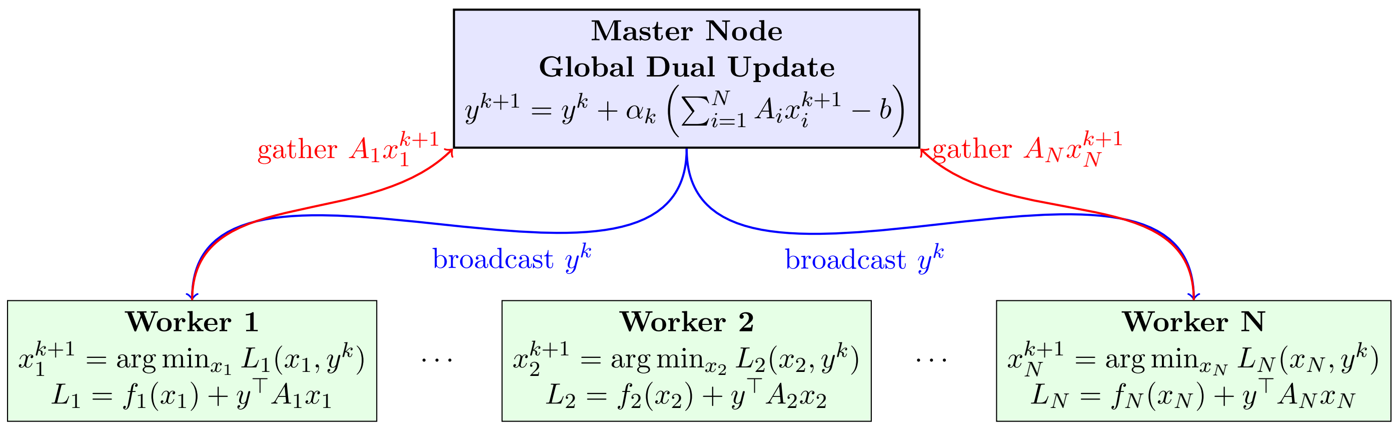
$$x_i^{k+1} := \arg\min_{x_i} L_i(x_i, y^k), \tag{10}$$

$$y^{k+1} := y^k + \alpha_k (Ax^{k+1} - b), \tag{11}$$

where each x_i-update in Eqn. 10 is performed in parallel, and the dual variable y in Eqn. 11 is updated using the aggregated constraint residual.

Each iteration here involves gathering the contributions $A_i x_i^{k+1}$ to compute the residual, followed by broadcasting the updated y^{k+1} back to all agents. This structure makes dual decomposition particularly suitable for distributed optimization as depicted follows:

Dual Decomposition Architecture


Master Node
Global Dual Update
$y^{k+1} = y^k + \alpha_k \left(\sum_{i=1}^N A_i x_i^{k+1} - b\right)$
gather $A_1 x_1^{k+1}$
gather $A_N x_N^{k+1}$
broadcast y^k
broadcast y^k
Worker 1
$x_1^{k+1} = \arg\min_{x_1} L_1(x_1, y^k)$
$L_1 = f_1(x_1) + y^\top A_1 x_1$
$\cdots$
Worker 2
$x_2^{k+1} = \arg\min_{x_2} L_2(x_2, y^k)$
$L_2 = f_2(x_2) + y^\top A_2 x_2$
$\cdots$
Worker N
$x_N^{k+1} = \arg\min_{x_N} L_N(x_N, y^k)$
$L_N = f_N(x_N) + y^\top A_N x_N$

- **Master Node**: Maintains dual variables y, performs global updates
- **Worker Nodes**: Solve local subproblems in parallel using y^k
- **Broadcast Phase**: Master sends y^k to all workers simultaneously
- **Gather Phase**: Workers send their contributions back to master
- **Lagrangian**: $L_i(x_i, y) = f_i(x_i) + y^\top A_i x_i$ for each subproblem

Dual decomposition forms the theoretical foundation for modern distributed and parallel optimization techniques. First developed in the 1960s through the pioneering work of Dantzig-Wolfe [8] and Benders [9] on large-scale linear programs, it established the framework for decomposition methods. The method gained prominence in the 1980s through decentralized optimization research, notably Tsitsiklis' work on distributed consensus optimization [10, 11].

2.4 Augmented Lagrangian & Method of Multipliers

Augmented Lagrangian methods enhance dual ascent by improving convergence, even when f is not strictly convex or finite. The augmented Lagrangian for Problem 4 is defined as:

Augmented Lagrangian

$$L_p = \underbrace{f(x)}_{\text{OBJECTIVE FUNCTION}} + \underbrace{y^\top (Ax - b)}_{\text{LAGRANGIAN}} + \underbrace{\frac{\rho}{2}\|Ax - b\|_2^2}_{\text{AUGMENTED LAGRANGIAN}}$$

where $\rho > 0$ is the penalty parameter. This is equivalent to solving:

$$\textbf{minimize } f(x) + \tfrac{\rho}{2}\|Ax - b\|_2^2$$
$$\textbf{subject to } Ax = b,$$

since the added term vanishes when x is feasible.

The augmented dual function $g_\rho(y) = \inf_x L_\rho(x, y)$ is often differentiable, allowing gradient-based updates. The *method of multipliers* applies dual ascent to this modified problem:

$$x^{k+1} := \arg\min_x L_\rho(x, y^k), \tag{12}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} - b) \tag{13}$$

Assuming f is differentiable, the optimality condition yields:

$$\nabla f(x^{k+1}) + A^T y^{k+1} = 0$$

Hence, the iterates become dual feasible, and the residual $Ax^{k+1} - b$ vanishes over time.

While the method improves convergence over standard dual ascent, it loses decomposability — $L_\rho(x, y)$ is not separable even if f is. Thus, it does not support parallel updates over variable blocks.

2.5 Alternating Direction Method of Multipliers (ADMM)

The *Alternating Direction Method of Multipliers (ADMM)* merges the decomposability of dual ascent with the robust convergence of the method of multipliers. It is designed to solve problems of the form:

$$\begin{array}{rl} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array} \tag{14}$$

where $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{p\times m}$, and $c \in \mathbb{R}^p$. The functions f and g are assumed to be convex.

This problem is a special case of linearly constrained optimization, where the objective is separable across variables x and z. The augmented Lagrangian is defined as :

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2, \tag{15}$$

with penalty parameter $\rho > 0$.

ADMM performs alternating minimization over x and z, followed by a dual variable update:

$$x^{k+1} := \arg\min_x \; L_\rho(x, z^k, y^k), \tag{16}$$

$$z^{k+1} := \arg\min_z \; L_\rho(x^{k+1}, z, y^k), \tag{17}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \tag{18}$$

Unlike the method of multipliers, which jointly minimizes over (x, z), ADMM alternates between them—similar to a single Gauss-Seidel pass, hence the name "*Alternating Direction*". This alternating update is what enables decomposition when f or g are separable.

Key Insight: ADMM Combines the Best of Both Worlds

The **Alternating Direction Method of Multipliers (ADMM)** is designed to merge two powerful ideas in constrained optimization:

- **Decomposability from Dual Decomposition:** ADMM allows the optimization problem to be split into subproblems that can be solved independently and in parallel when the objective is separable. This makes it ideal for distributed and large-scale systems.
- **Robust Convergence from the Method of Multipliers:** By introducing an augmented Lagrangian with a quadratic penalty, ADMM inherits superior convergence properties even in the absence of strict convexity or smoothness.

In summary, ADMM provides a practical and scalable approach that combines the algorithmic simplicity and parallelizability of dual decomposition with the convergence reliability of the method of multipliers.

In practice, a scaled version of the algorithm is often used. Letting $u = y/\rho$, the updates become:

$$x^{k+1} := \arg\min_x \; f(x) + \frac{\rho}{2}\|Ax + Bz^k - c + u^k\|_2^2, \tag{19}$$

$$z^{k+1} := \arg\min_z \; g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - c + u^k\|_2^2, \tag{20}$$

$$u^{k+1} := u^k + Ax^{k+1} + Bz^{k+1} - c. \tag{21}$$

The algorithm state is represented by (z^k, y^k), while x^k is computed internally. If there is a swapping of x with z, and f, A with g, B, a valid variant can be obtained with reversed update order. Though nearly symmetric, the order of x and z matters due to the timing of the dual update.

3 Applications of ADMM

In this section, the application and feasibility of the Alternating Direction Method of Multipliers (ADMM) is described across various domains. ADMM is widely used due to its decomposition capabilities and convergence properties for large-scale convex optimization problems. The application of ADMM was analyzed in the following three scenarios:

- **Case 1:** ADMM in solving the LASSO (Least Absolute Shrinkage and Selection Operator) problem
- **Case 2:** Image denoising or reconstruction using ADMM under a LASSO formulation
- **Case 3:** ADMM for hybrid precoding in MIMO systems

3.1 Case 1: ADMM in LASSO

The LASSO optimization problem aims to minimize a least-squares loss function with an ℓ_1-norm regularization to promote sparsity in the solution. ADMM efficiently solves this problem by splitting the objective into separable components and alternating between minimization steps and dual updates. It is particularly effective for high-dimensional regression and feature selection tasks. The LASSO problem promotes sparsity in the solution by combining a least-squares loss with an ℓ_1 regularization term. The basic LASSO problem is defined as:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1 \tag{22}$$

where:

- $A \in \mathbb{R}^{m\times n}$ is the measurement or feature matrix,
- $b \in \mathbb{R}^m$ is the observed data,
- $x \in \mathbb{R}^n$ is the variable to estimate,
- $\lambda > 0$ controls the trade-off between fidelity and sparsity.

To apply ADMM, an auxiliary variable z is introduced and thus the problem can be reformulated as:

$$\min_{x,z} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|z\|_1 \quad \text{subject to} \quad x = z \tag{23}$$

the constraint is equivalent to Ax + Bz = c, The general form of the augmented Lagrangian used by ADMM as in Eqn 15 is:.

$$\mathcal{L}_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \tag{24}$$

For the LASSO problem, the components are:

- $f(x) = \frac{1}{2}\|Ax - b\|_2^2$
- $g(z) = \lambda\|z\|_1$
- $A = I, B = -I, \ c = 0$

Substituting into the Lagrangian gives:

$$\mathcal{L}_\rho(x, z, y) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|z\|_1 + y^T(x - z) + \frac{\rho}{2}\|x - z\|_2^2 \tag{25}$$

Using a scaled dual variable $u = y/\rho$, the expression becomes:

$$\mathcal{L}_\rho(x, z, u) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|z\|_1 + \frac{\rho}{2}\|x - z + u\|_2^2 \tag{26}$$

This formulation allows the problem to be solved via alternating minimization over x and z, with an update of u at each iteration, enabling convergence to a sparse and accurate solution.

The ADMM iterations consist of three main steps:

***x*-update:** **Solve for x^{k+1}**

$$x^{k+1} = \arg\min_x \left(\frac{1}{2}\|Ax - b\|_2^2 + \frac{\rho}{2}\|x - z^k + u^k\|_2^2 \right) \tag{27}$$

The objective function is expanded as such:

$$\begin{aligned} f(x) &= \frac{1}{2}(Ax-b)^T(Ax-b) + \frac{\rho}{2}(x - z^k + u^k)^T(x - z^k + u^k) \quad (28) \\ &= \frac{1}{2}x^T A^T A x - x^T A^T b + \frac{1}{2}b^T b + \frac{\rho}{2}x^T x - \rho x^T(z^k - u^k) + \text{const.} \quad (29) \end{aligned}$$

Ignoring constants, f(x) differentiated with respect to x as:

$$\nabla_x f(x) = A^T A x - A^T b + \rho x - \rho(z^k - u^k) \tag{30}$$

Setting the derivative to zero:

$$A^T A x + \rho x = A^T b + \rho(z^k - u^k) \tag{31}$$

Solving for x:

$$x^{k+1} = (A^T A + \rho I)^{-1}(A^T b + \rho(z^k - u^k)) \tag{32}$$

***z*-update:** **Solve for z^{k+1}**

Given the LASSO reformulated problem:

$$\min_{x,z} \ \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|z\|_1 \quad \text{subject to} \quad x = z \tag{33}$$

After forming the augmented Lagrangian, the z-update step is:

$$z^{k+1} = \arg\min_z \left(\lambda\|z\|_1 + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2 \right) \tag{34}$$

This is a standard proximal operator for the ℓ_1-norm.

Intuition:

- $\lambda\|z\|_1$ encourages sparsity in z.
- $\frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2$ penalizes the difference between z and $x^{k+1} + u^k$.

Together, these lead to a soft-thresholding operation.

Solution: Define:

$$v = x^{k+1} + u^k \tag{35}$$

Then:

$$z^{k+1} = \arg\min_z \left(\lambda\|z\|_1 + \frac{\rho}{2}\|z - v\|_2^2 \right) \tag{36}$$

This yields the well-known soft-thresholding solution:

$$z^{k+1} = S_{\lambda/\rho}(v) = S_{\lambda/\rho}(x^{k+1} + u^k) \tag{37}$$

where the soft-thresholding operator $S_\theta(v)$ is applied element-wise:

$$S_\theta(v_i) = \text{sign}(v_i) \cdot \max(|v_i| - \theta, 0) \tag{38}$$

The z-update in ADMM for LASSO is a soft-thresholding step:

- It shrinks the elements of $x^{k+1} + u^k$ toward zero.
- Elements below the threshold λ/ρ are set to zero.
- Others are reduced in magnitude by λ/ρ, preserving their sign.

u-update: **Update the scaled dual variable**

$$u^{k+1} = u^k + x^{k+1} - z^{k+1} \tag{39}$$

The dual variable u accumulates the discrepancy between x and z over iterations. This step is critical in enforcing the constraint $x = z$ gradually through penalization.

- When x and z are different, u adjusts to penalize the gap.
- Over iterations, this update pushes x and z closer.
- This is what makes ADMM effective in enforcing constraints, even in distributed settings.

These iterative steps, as summarized in Algorithm 1, continue until convergence criteria are met, effectively solving the original LASSO problem with guaranteed convergence properties under convexity assumptions.

Algorithm 1: Solving LASSO via ADMM

Input: Data matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, observation vector $\mathbf{b} \in \mathbb{R}^m$, regularization parameter λ, ADMM penalty parameter ρ, maximum iterations K, convergence threshold ϵ.

1. **Initialize:** Set $\mathbf{x}^{(0)} = \mathbf{0}$, $\mathbf{z}^{(0)} = \mathbf{0}$, $\mathbf{u}^{(0)} = \mathbf{0}$ (all in $\mathbb{R}^n$).
2. **Repeat for** $k = 0$ **to** $K - 1$**:**
   (a) **x-update:**
   $\mathbf{x}^{(k+1)} = (\mathbf{A}^T\mathbf{A} + \rho\mathbf{I})^{-1}(\mathbf{A}^T\mathbf{b} + \rho(\mathbf{z}^{(k)} - \mathbf{u}^{(k)}))$
   (b) **z-update :** $\mathbf{z}^{(k+1)} = \mathcal{S}_{\lambda/\rho}(\mathbf{x}^{k+1} + \mathbf{u}^k)$
   (c) **u-update (dual variable):**
   $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}$
   (d) **Compute residuals:**
   $r^{(k+1)} = \|\mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}\|_2$
   $s^{(k+1)} = \rho \cdot \|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\|_2$
   (e) **Check convergence:** If $r^{(k+1)}_{primal} < \epsilon$ and $r^{(k+1)}_{dual} < \epsilon$, break.
3. **Output:** Solution vectors $\mathbf{x}^\star = \mathbf{x}^{(k+1)}$, $\mathbf{z}^\star = \mathbf{z}^{(k+1)}$, $\mathbf{u}^\star = \mathbf{u}^{(k+1)}$.

Simulation Parameters

To validate the effectiveness of ADMM for LASSO, a MATLAB implementation was developed. The setup defines $N_x = 1000$ predictors and $N_b = 1000$ samples to solve a sparse regression problem. A synthetic dataset is generated using a random Gaussian matrix $A \in \mathbb{R}^{1000\times1000}$ and a sparse ground-truth coefficient vector `true_coeffs` $\in \mathbb{R}^{1000}$. The observed vector b is formed by $A \cdot$ `true_coeffs` with added Gaussian noise.

This models a noisy linear system where the objective is to recover sparse coefficients from corrupted measurements. All variables (x, z, and y) are initialized to zero, and matrix products A^TA and A^Tb are precomputed to improve performance. The core ADMM algorithm alternates between updating the primal variable x, the auxiliary variable z using soft-thresholding, and the dual variable u. Convergence is monitored via primal and dual residuals.

Experiments include: (i) varying ρ with fixed λ, (ii) varying λ with fixed ρ, and (iii) visualizing convergence under fixed parameters. This simulates real-world regression scenarios like identifying key features in high-dimensional noisy data, highlighting ADMM's scalability and efficiency in sparse recovery.

Use Case Example: Stock Market Prediction Consider a stock price prediction scenario, where a model is developed using 1,000 distinct indicators as input features, such as technical metrics, sentiment scores, macroeconomic factors, and trading volumes. Out of these, only a handful (perhaps 10–20) are actually influential. In this context:

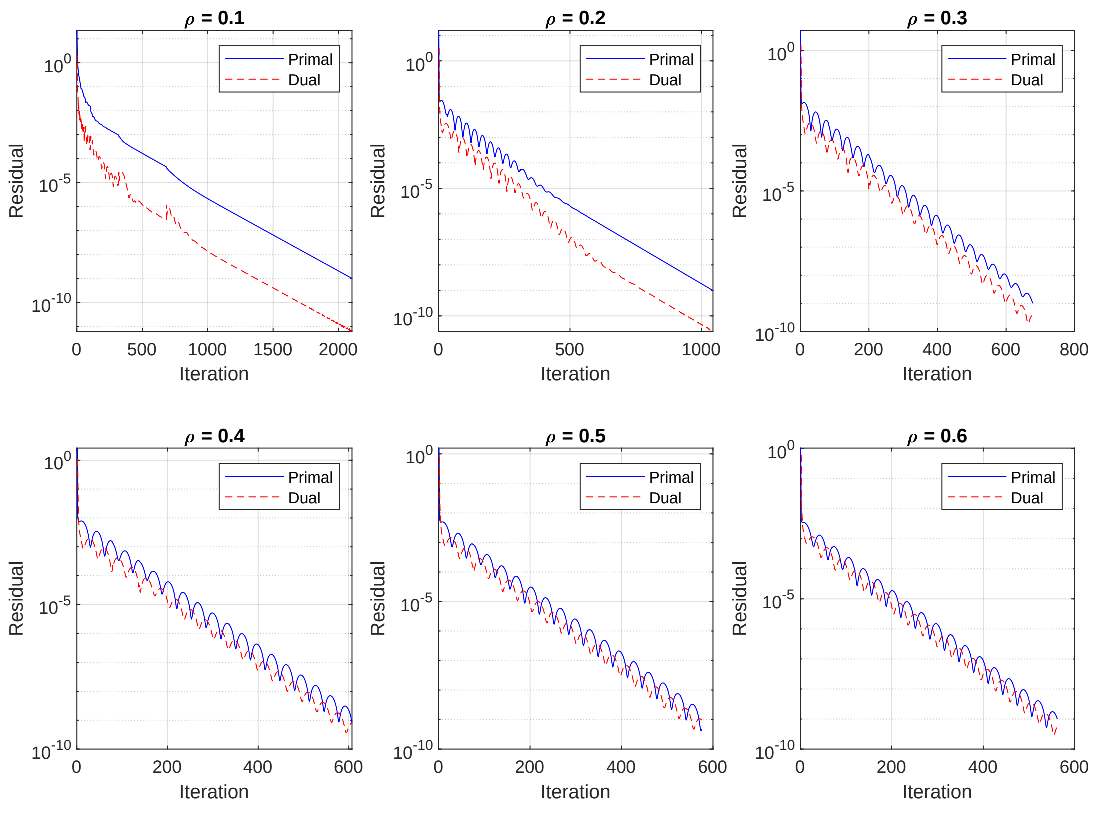
- `true_coeffs` represents the true but unknown weights of these influential indicators.
- The matrix A contains daily observations of all 1000 indicators.
- The response vector b contains the observed stock returns or price changes, corrupted by market noise.

Using ADMM to solve the LASSO problem, the aim here is to identify the few relevant indicators (i.e., non-zero entries in x) that are most predictive of the stock's movement, while ignoring irrelevant or noisy features.

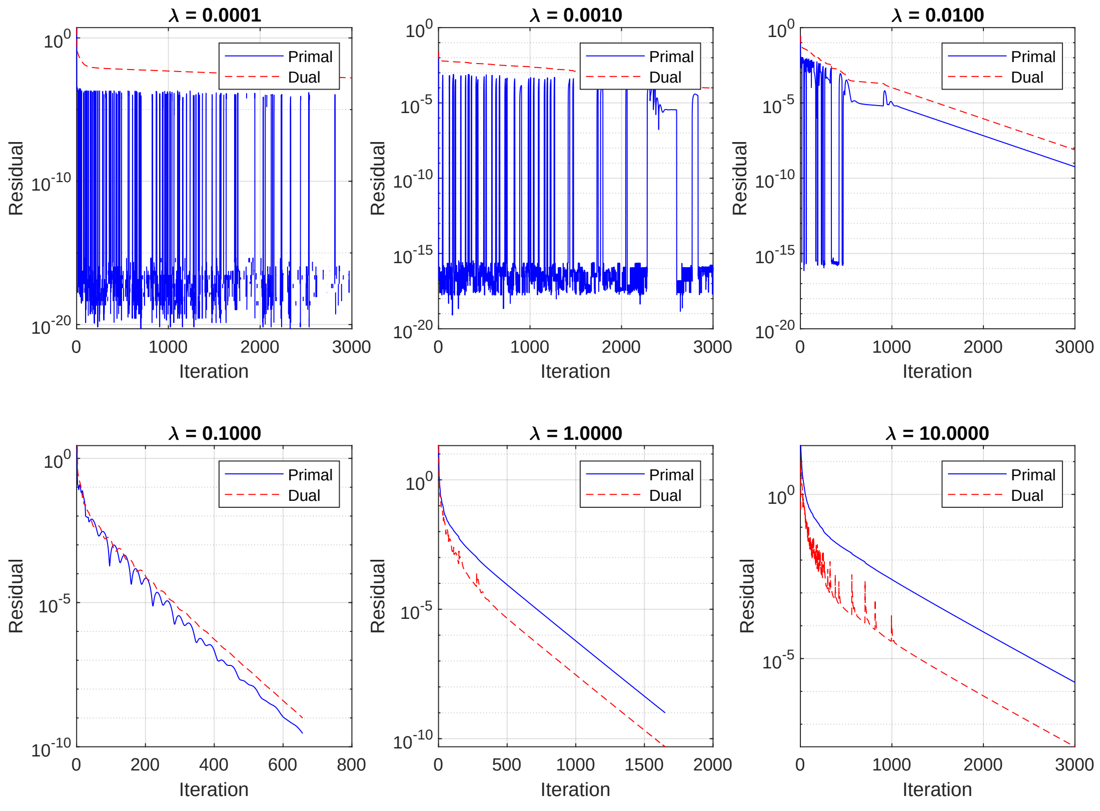
MATLAB Simulation Parameters

Parameter	**Value / Description**
Number of predictors (N_x)	1000
Number of observations (N_b)	1000
Max ADMM iterations	3600
Convergence threshold (ϵ)	10^{-13}
Measurement matrix (A)	$\mathcal{N}(0,1)$ (Gaussian random): `A = randn(N_b, N_x)`
True coefficients (`true_coeffs`)	$\mathcal{N}(0,1)$: `true_coeffs = randn(N_x, 1)`
Observations (b)	$A \cdot$ `true_coeffs`+ noise, where noise $\sim \mathcal{N}(0, 0.5^2)$:

Table 1: Parameter settings for synthetic LASSO simulation in MATLAB

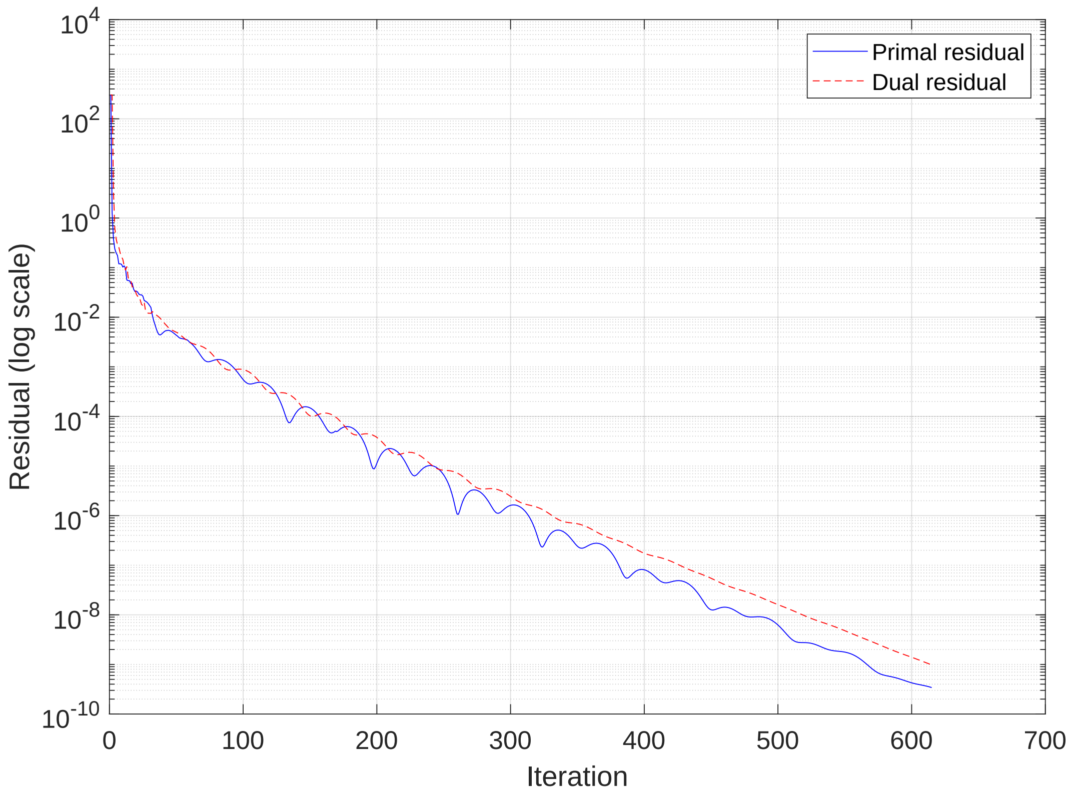


(a) ADMM Convergence for Varying ρ ($\lambda = 0.1$)

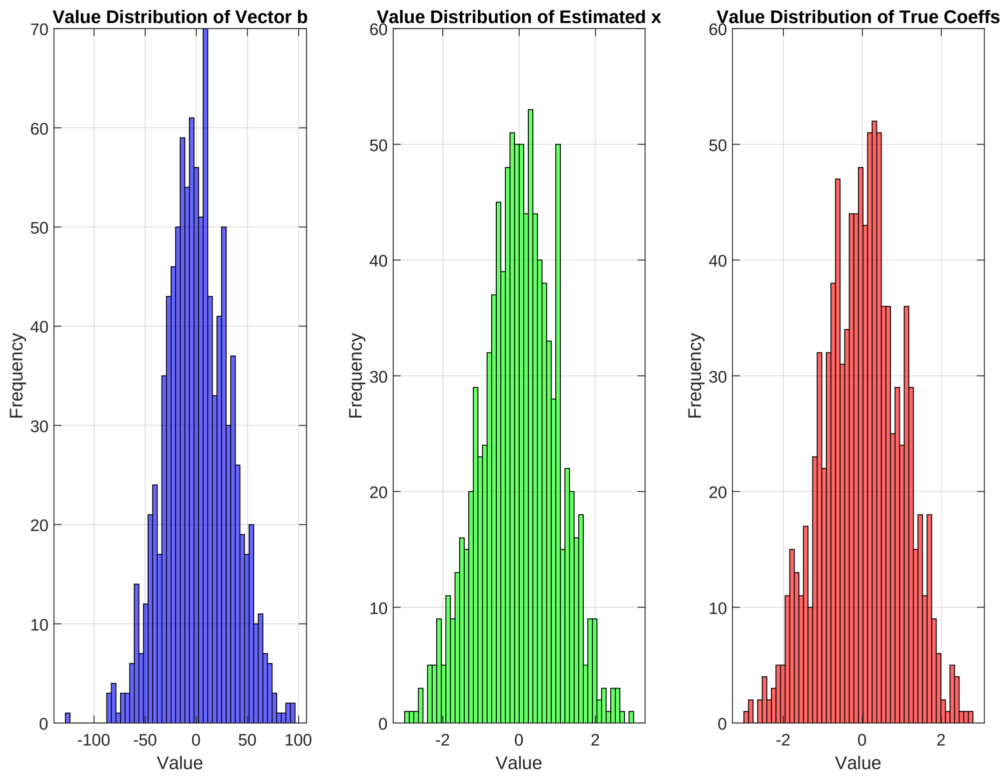


(b) ADMM Convergence for Varying λ ($\rho = 1$)

Figure 1: Effect of varying ρ and λ on ADMM convergence

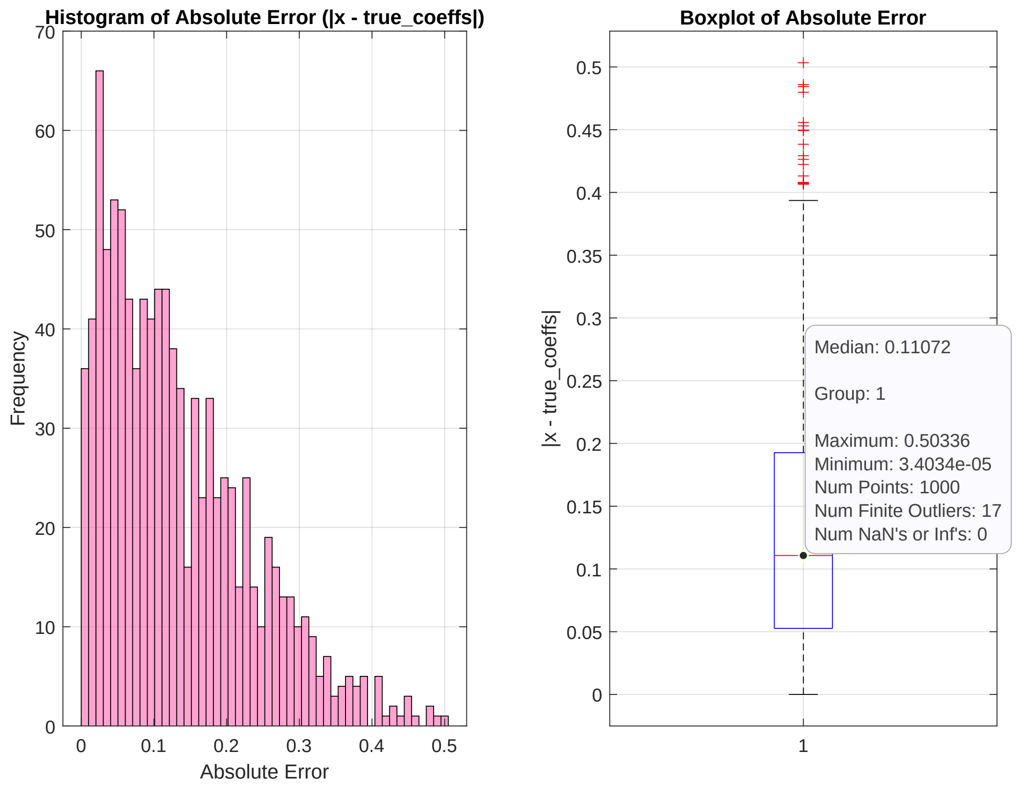


(a) Fixed $\rho = 1$, $\lambda = 0.1$

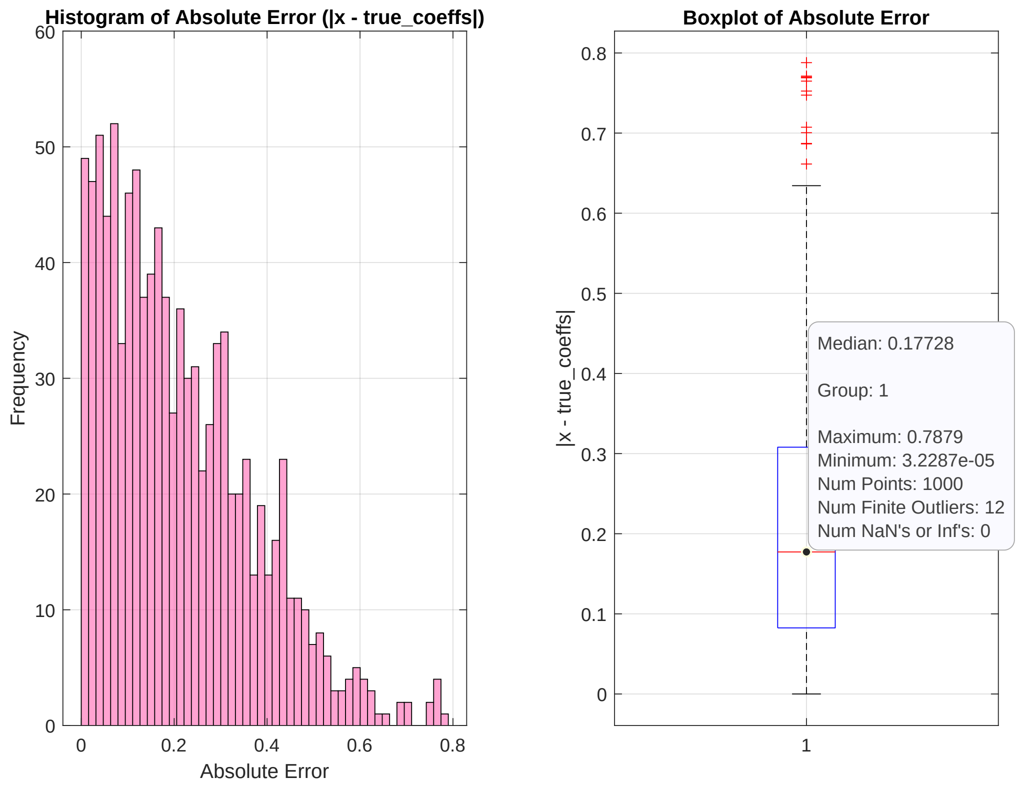


(b) Distribution of b, estimated x, and true coefficients

Figure 2: Performance visualization and parameter distribution



(a) Absolute Error Distribution: $\lambda = 0.1$



(b) Absolute Error Distribution: $\lambda = 0.001$

Figure 3: Error distributions under different parameters

Result Analysis

This section presents the empirical evaluation of the ADMM-based LASSO algorithm under various configurations of the penalty parameter ρ and the regularization weight λ. Convergence behavior, parameter distributions, and error characteristics are visualized here to assess the algorithm's stability and robustness.

Figure 1(a) displays the evolution of primal and dual residuals for different values of ρ ranging from 0.1 to 0.6. As ρ increases, the convergence speed initially improves but eventually plateaus. Specifically, $\rho = 0.2$ and $\rho = 0.3$ offer faster convergence within fewer iterations, balancing the trade-off between primal feasibility and dual stability. This highlights the importance of tuning ρ for optimal performance.

Figure 1(b) explores the impact of changing λ on convergence. A small λ (e.g., 0.0001) leads to slower convergence, as the sparsity penalty is weak and the solution remains closer to least-squares. Larger λ values (e.g., 0.1 or 1.0) enforce more sparsity and stabilize the optimization faster. This reinforces that λ acts as a lever to control sparsity in the solution vector.

As shown in Figure 2(a), fixing both ρ and λ demonstrates steady convergence of both residuals. This configuration is used in subsequent visualizations to assess the estimated coefficients and error distributions. Figure 2(b) presents the value distributions of the observed response vector b, the estimated coefficients x, and the ground-truth sparse vector `true_coeffs`. The estimated x closely mirrors the sparsity pattern of the ground truth, demonstrating the ADMM algorithm's ability to recover underlying signals despite the presence of noise in b.

Error Analysis

Figures 3(a) and 3(b) depict the distribution of absolute errors $|x - \texttt{true\_coeffs}|$ under two different regularization levels: $\lambda = 0.1$ and $\lambda = 0.001$.

- **Low Regularization ($\lambda = 0.001$):** Figure 3(b) shows that many estimated coefficients deviate from ground truth, as indicated by a wider error spread and higher median error (≈ 0.177). The solution is less sparse and more sensitive to noise.
- **Moderate Regularization ($\lambda = 0.1$):** Figure 3(a) demonstrates improved accuracy. The histogram shows tighter concentration of absolute errors near zero, and the median error drops to approximately 0.11. The corresponding boxplot reveals fewer extreme outliers and a more compact interquartile range, confirming the advantage of balanced regularization.

These results confirm that the ADMM approach is highly sensitive to the choice of ρ and λ. Moderate values of ρ (0.2–0.4) and λ (0.1) consistently yield fast convergence and high-quality sparse estimates. The method effectively separates signal from noise and is suitable for high-dimensional regression problems such as feature selection in stock market prediction or biomedical signal analysis.

3.2 Case 2: Image Reconstruction Using ADMM for LASSO

In image processing, ADMM can be applied to LASSO-like formulations to perform image reconstruction. Here, the goal is to recover a clean image from noisy measurements by penalizing differences with sparse representations. ADMM alternates between solving a smooth data fidelity term and a nonsmooth sparsity-inducing regularizer, enabling efficient convergence while preserving image structure.

When applying LASSO to image reconstruction using ADMM, it is assumed that $A = I$ (identity matrix), which models a simple case where the noisy observation is a direct (but corrupted) version of the signal. The LASSO objective becomes:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - b\|_2^2 + \lambda \|x\|_1$$

Now an auxiliary variable z is introduced, allowing the problem to be reformulated as follows:

$$\min_{x,z} \frac{1}{2}\|x - b\|_2^2 + \lambda \|z\|_1 \quad \text{subject to} \quad x = z$$

Augmented Lagrangian and ADMM for Identity A

The augmented Lagrangian is:

$$\mathcal{L}_\rho(x, z, u) = \frac{1}{2}\|x - b\|_2^2 + \lambda \|z\|_1 + \frac{\rho}{2}\|x - z + u\|_2^2$$

The scaled form of the dual variable is employed here, defined as $u = y/\rho$. .

ADMM Updates for $A = I$ (Image Reconstruction)

Repeat the following steps until convergence:

1. x**-update:**

$$x^{k+1} = \arg\min_x \left(\frac{1}{2}\|x - b\|_2^2 + \frac{\rho}{2}\|x - z^k + u^k\|_2^2 \right)$$

Taking the gradient and setting it to zero:

$$(1 + \rho)x = b + \rho(z^k - u^k) \quad \Rightarrow \quad x^{k+1} = \frac{1}{1 + \rho}(b + \rho(z^k - u^k))$$

2. z**-update:** (soft-thresholding)

$$z^{k+1} = \arg\min_z \left(\lambda \|z\|_1 + \frac{\rho}{2}\|x^{k+1} - z + u^k\|_2^2 \right)$$

This is solved by the soft-thresholding operator:

$$z^{k+1} = S_{\lambda/\rho}(x^{k+1} + u^k), \quad \text{where} \quad S_\theta(v) = \text{sign}(v) \cdot \max(|v| - \theta, 0)$$

3. u**-update:**

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}$$

This formulation matches the MATLAB code where $A = I$, and updates are simple element-wise operations applied to the image vector, as summarized in Algorithm 2.

Algorithm 2 : Image Reconstruction Using ADMM for LASSO

Input: Noisy grayscale image b, regularization parameter λ, ADMM penalty parameter ρ, maximum iterations K, convergence threshold ϵ.

```
1. Initialize: Set $x = 0$, $z = 0$, $y = 0$ (vectors the size of the image).
2. Repeat for $k = 1$ to $K$:
    (a) $x$-update:
        $x^{(k+1)} = \frac{1}{1+\rho}(b + \rho(z^{(k)} - y^{(k)}))$
    (b) $z$-update (soft thresholding):
        $x_{\text{hat}} = x^{(k+1)} + y^{(k)}$
        $z^{(k+1)} = \text{sign}(x_{\text{hat}}) \cdot \max(|x_{\text{hat}}| - \lambda/\rho, 0)$
    (c) $y$-update (dual variable):
        $y^{(k+1)} = y^{(k)} + x^{(k+1)} - z^{(k+1)}$
    (d) Compute residuals:
        $r^{(k)} = \|x^{(k+1)} - z^{(k+1)}\|_2, \quad s^{(k)} = \rho \cdot \|z^{(k+1)} - z^{(k)}\|_2$
    (e) Check convergence: If $r^{(k)} < \epsilon$ and $s^{(k)} < \epsilon$, break.
3. Reshape: $x^{(k+1)}$ to image format for visualization.
4. Output: Denoised image, residual plots, error map, and metrics (PSNR, SSIM, $\|\text{Error}\|_2$).
```

Simulation Parameters

To assess the effectiveness of ADMM in reconstruction and sparse recovery tasks, experiments are conducted using a standard grayscale image corrupted with Gaussian noise. Specifically, the well-known "Cameraman" image is utilized, converted it to a double-precision grayscale format, and vectorized it into a one-dimensional array representing the true image signal. To simulate realistic degradation, zero-mean Gaussian noise is added with a standard deviation of 0.3, resulting in a noisy observation vector. In this setup, the measurement matrix A is the identity matrix, implying a direct observation model where each pixel is independently corrupted. The resulting optimization problem becomes a LASSO formulation that seeks to recover a denoised version of the image while enforcing sparsity in the pixel domain. Key simulation parameters included a range of regularization weights λ (from 0.1 to 5.0), penalty parameter ρ values (from 0.1 to 100.0), a maximum iteration cap of 3000, and a convergence threshold set to 10^{-9}. This setup allows to systematically explore the interplay between regularization strength, convergence behavior, and reconstruction fidelity, with metrics such as PSNR, SSIM, and ℓ_2-norm error guiding the evaluation.

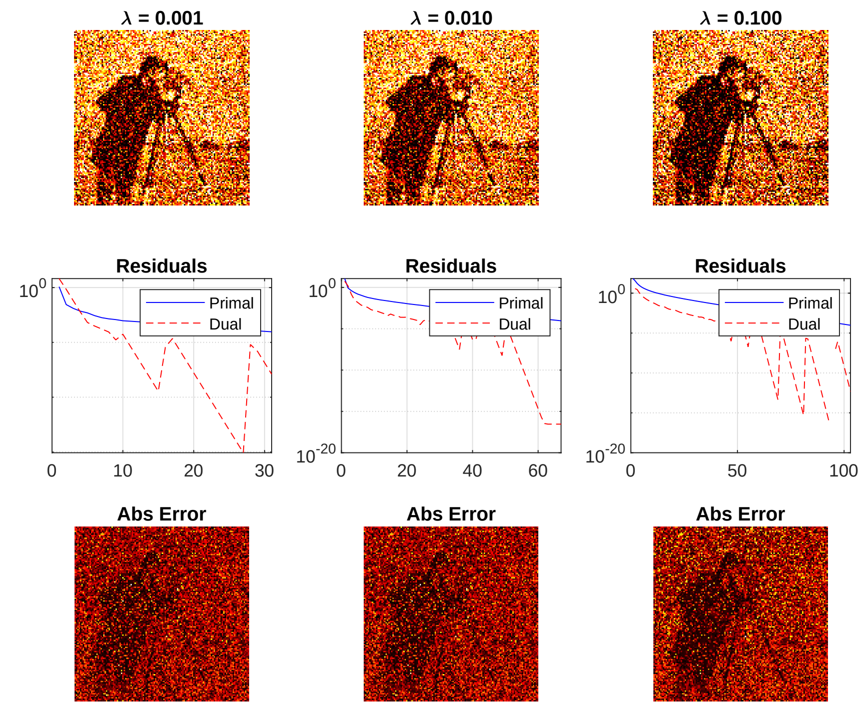


Figure 4: Reconstruction via LASSO-ADMM for Varying λ ($\rho = 0.01$)

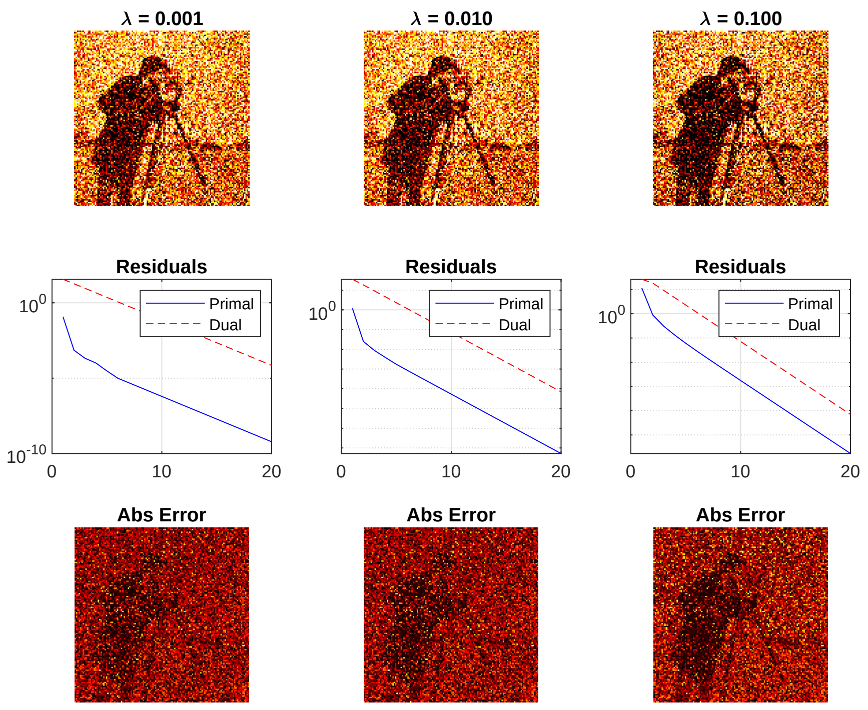


Figure 5: Reconstruction via LASSO-ADMM for Varying λ ($\rho = 0.1$)

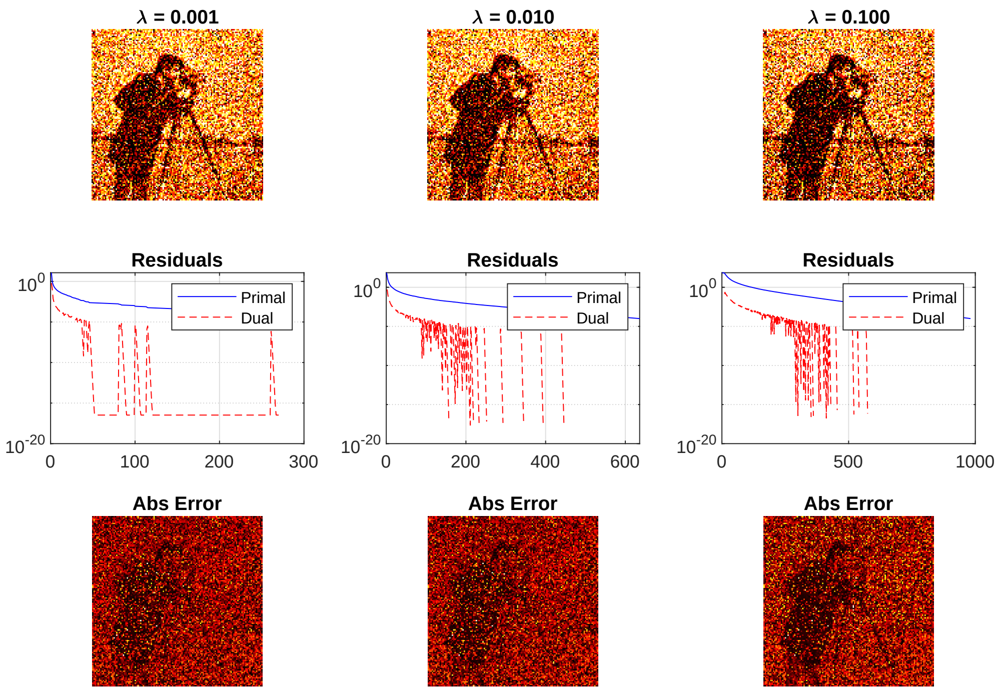


Figure 6: Reconstruction via LASSO-ADMM for Varying λ ($\rho = 1$)

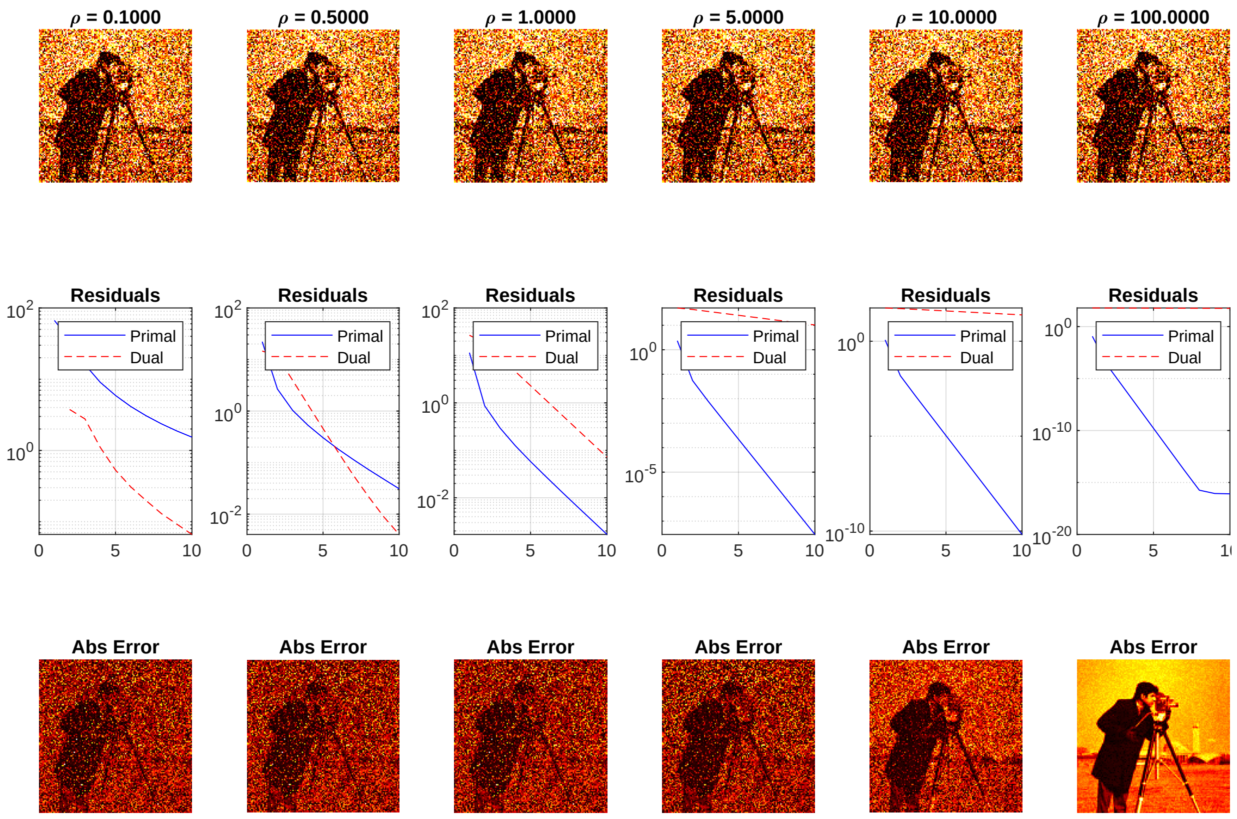


Figure 7: Reconstruction via LASSO-ADMM for Varying ρ ($\lambda = 0.1$)

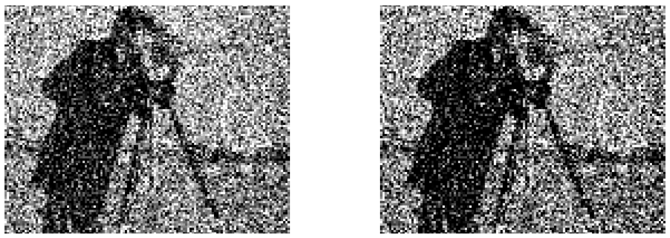


Figure 8: Reconstruction via LASSO-ADMM for Varying ρ ($\lambda = 0.1$)

Result Analysis

In this section, the image reconstruction performance of the ADMM-based LASSO algorithm under varying configurations of regularization parameter λ and penalty parameter ρ is analyzed. The results are visualized through reconstructed image snapshots and plotted residuals, providing qualitative and quantitative insights into the reconstruction effectiveness.

Figure 4–6 illustrate the effect of changing λ values when ρ is fixed at 0.01, 0.1, and 1.0, respectively. As observed, for small values of λ (e.g., 0.001 or 0.01), the reconstructed images retain noise, reflecting under-regularization. Increasing λ gradually improves reconstruction until a certain threshold, beyond which oversmoothing occurs—resulting in loss of structural details. This aligns with theory: small λ allows noisy coefficients to survive, while large λ

suppresses both noise and important signal components. Conversely, Figure 8 presents the impact of varying ρ for a fixed $\lambda = 0.1$. Moderate values of ρ (such as 0.1 and 1) strike a good balance between primal and dual residual decay, leading to stable convergence. Too large ρ (e.g., 100) introduces instability in the dual update and results in poor estimates. The residual plots confirm this—showing higher residuals and slower convergence for extreme ρ values. Finally, Figure 8 (right) overlays the reconstructed images across configurations. Visually, it becomes apparent that the best image quality is achieved with $\lambda = 0.1$ and ρ around 0.1–1.

Observations from Experimental Results

Definition of Performance Matrix- PSNR and SSIM

PSNR (Peak Signal-to-Noise Ratio) is a logarithmic measure of the peak error between two images. It is defined as:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right), \quad \text{where} \quad \text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

Here, x_i is the pixel in the original image and $\hat{x}_i$ is the corresponding pixel in the reconstructed image. Higher PSNR indicates better reconstruction quality.

Note: MAX is the maximum possible pixel value. For 8-bit grayscale images, MAX = 255. For normalized images in [0,1], MAX = 1.

SSIM (Structural Similarity Index) quantifies image similarity based on structural information, luminance, and contrast. It is defined as:

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)}$$

where μ_x, $\mu_{\hat{x}}$ are the mean intensities, σ_x^2, $\sigma_{\hat{x}}^2$ are variances, and $\sigma_{x\hat{x}}$ is the covariance. Constants C_1, C_2 stabilize division. SSIM ranges from 0 to 1; 1 indicates perfect similarity.

Effect of Varying λ for Fixed $\rho = 0.1$

λ	PSNR (dB)	SSIM	$\|\text{Error}\|_2$
0.1	11.81	0.1533	32.87
0.5	7.82	0.0786	52.02
1.0	5.61	0.0035	67.08
5.0	5.61	0.0035	67.08

Table 2: Image reconstruction metrics for varying λ with fixed $\rho = 0.1$

As observed from Table 2, varying the regularization parameter λ significantly influences the quality of image reconstruction. When $\lambda = 0.1$, the model strikes a desirable balance between preserving important image structures and removing noise, yielding the highest PSNR (11.81 dB) and the best SSIM (0.1533). However, as λ increases, the regularization term dominates, enforcing excessive sparsity in the solution. This results in underfitting, where many informative components of the image are suppressed, leading to substantial loss of detail. This degradation is reflected in the sharp drop in PSNR and SSIM values, with both metrics plateauing at low values for $\lambda \geq 1.0$. In particular, the SSIM nearly vanishes at high λ, confirming severe structural distortion. Therefore, a small but non-negligible value of λ is essential for maintaining both fidelity and sparsity in the reconstructed image.

Effect of Varying ρ for Fixed $\lambda = 0.1$

ρ	PSNR (dB)	SSIM	$\|\text{Error}\|_2$
0.1	11.83	0.1509	32.80
0.5	11.77	0.1501	33.03
1.0	11.77	0.1501	33.02
5.0	11.72	0.1574	33.21
10.0	10.49	0.1570	38.24
100.0	6.31	0.0845	61.92

Table 3: Image reconstruction metrics for varying ρ with fixed $\lambda = 0.1$

Interpretation: From Table 3, it is observed that the convergence and reconstruction performance of ADMM is sensitive to the penalty parameter ρ. When ρ is within a moderate range ($0.1 \leq \rho \leq 5.0$), the PSNR and SSIM remain stable, and the ℓ_2 error norm stays low, indicating that the algorithm is effectively balancing the fit to the data and regularization. As ρ increases beyond this optimal range, particularly at $\rho = 10.0$ and $\rho = 100.0$, performance starts to deteriorate. A high ρ leads to an overemphasis on enforcing the $x = z$ constraint too aggressively, which destabilizes the updates and causes convergence to an inaccurate solution. This is evident from the sharp rise in the ℓ_2 error norm and the decline in PSNR and SSIM. In practice, choosing a small to moderate ρ ensures stable and accurate recovery, while overly large values can introduce numerical instability and suboptimal reconstructions.

Joint Effect of λ and ρ

The analysis across all experiments reveals a consistent pattern: optimal reconstruction quality is achieved when λ and ρ are tuned to moderate values. Extremely small values of λ fail to suppress noise effectively, leading to under-regularized and noisy outputs. Conversely, overly large values of λ result in excessive sparsity, which strips away meaningful signal components and leads to high distortion, as shown by plummeting PSNR and SSIM scores. Similarly, moderate values of ρ provide a stable convergence trajectory for ADMM by properly balancing the primal and dual updates. However, very large ρ values can distort this balance, driving the solution away from optimality due to aggressive penalization in consensus updates. Therefore, the recommended operating ranges for best performance are $\lambda \approx 0.01$ to 0.1 and $\rho \approx 0.1$ to 5.0. The most robust results were consistently observed around $\lambda = 0.1$ and $\rho = 0.1$.

Parameter	Optimal Range
λ	∼0.01 to 0.1
ρ	∼0.1 to 5.0

3.3 Case 3: ADMM in Hybrid Precoding for MIMO Systems

Hybrid precoding in millimeter-wave MIMO systems poses a non-convex optimization challenge due to unit-modulus constraints. ADMM provides a powerful iterative framework to decompose this problem into manageable subproblems: one for analog precoder update and another for digital precoder estimation. This method ensures convergence while respecting the hardware constraints inherent in analog RF components.

As a motivating application, the design of hybrid precoders in a mmWave MIMO system is considered. Hybrid beamforming seeks to approximate a fully digital precoder $\mathbf{F}_{\text{opt}}$ with a product of analog and digital components:

$$\mathbf{F} = \mathbf{F}_{\text{RF}}\mathbf{F}_{\text{BB}} \tag{40}$$

subject to unit-modulus constraints on $\mathbf{F}_{\text{RF}}$ and power constraints on $\mathbf{F}_{\text{BB}}$.

The precoder design can be posed as an optimization problem as:

$$\min_{\mathbf{F}_{\text{RF}},\mathbf{F}_{\text{BB}}} \|\mathbf{F}_{\text{opt}} - \mathbf{F}_{\text{RF}}\mathbf{F}_{\text{BB}}\|_F^2 \quad \text{s.t. } |(\mathbf{F}_{\text{RF}})_{i,j}| = 1,\ \|\mathbf{F}_{\text{RF}}\mathbf{F}_{\text{BB}}\|_F^2 \le P \tag{41}$$

This can be decomposed and solved via ADMM, as summarized later in Algorithm 3.

Problem Formulation

Given the optimal fully-digital precoder $\mathbf{F}_{\text{opt}} \in \mathbb{C}^{N_t \times KN_s}$, the hybrid precoder is designed to minimize the Euclidean distance:

$$\min_{\mathbf{F}_R,\mathbf{F}_B} \|\mathbf{F}_{\text{opt}} - \mathbf{F}_R\mathbf{F}_B\|_F^2 \tag{42}$$

subject to:

$$|\mathbf{F}_R(i,j)| = 1\ \forall i,j \tag{43}$$

$$\|\mathbf{F}_R\mathbf{F}_B\|_F^2 = KN_s \tag{44}$$

Alternating Optimization Strategy

To solve the problem, alternate between solving:

Baseband Precoder Update

Given $\mathbf{F}_R$, solve:

$$\min_{\mathbf{F}_B} \|\mathbf{F}_{\text{opt}} - \mathbf{F}_R\mathbf{F}_B\|_F^2 \quad \text{s.t.} \quad \|\mathbf{F}_R\mathbf{F}_B\|_F^2 = KN_s \tag{45}$$

Closed-form solution:

$$\mathbf{F}_B = (\mathbf{F}_R^H \mathbf{F}_R)^{-1} \mathbf{F}_R^H \mathbf{F}_{\text{opt}}, \quad \mathbf{F}_B \leftarrow \sqrt{\frac{KN_s}{\|\mathbf{F}_R \mathbf{F}_B\|_F^2}} \cdot \mathbf{F}_B \tag{46}$$

Derivation of ADMM for Hybrid Precoding Design

To solve the RF precoder optimization subproblem, consider the objective:

$$\min_{\mathbf{F}_R} \|\mathbf{F}_{\text{opt}} - \mathbf{F}_R \mathbf{F}_B\|_F^2 \quad \text{s.t. } |\mathbf{F}_R(i,j)| = 1 \tag{47}$$

Vectorizing this expression gives:

$$\begin{aligned} \|\mathbf{F}_{\text{opt}} - \mathbf{F}_R \mathbf{F}_B\|_F^2 &= \|\text{vec}(\mathbf{F}_{\text{opt}} - \mathbf{F}_R \mathbf{F}_B)\|_2^2 \quad (48) \\ &= \|\text{vec}(\mathbf{F}_{\text{opt}}) - (\mathbf{F}_B^T \otimes I_{N_t})\text{vec}(\mathbf{F}_R)\|_2^2 \quad (49) \end{aligned}$$

Define:

$$\begin{aligned} \mathbf{y} &= \text{vec}(\mathbf{F}_{\text{opt}}) \\ \mathbf{w} &= \text{vec}(\mathbf{F}_R) \\ \mathbf{A} &= \mathbf{F}_B^T \otimes I_{N_t} \end{aligned}$$

So the problem becomes:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{A}\mathbf{w}\|_2^2 \quad \text{s.t. } |w_i|^2 = 1 \quad \forall i \tag{50}$$

The problem is homogenized by introducing a scalar auxiliary variable $t \in \mathbb{C}$, and define:

$$\bar{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ t \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \mathbf{A}^H \mathbf{A} & -\mathbf{A}^H \mathbf{y} \\ -\mathbf{y}^H \mathbf{A} & \mathbf{y}^H \mathbf{y} \end{bmatrix} \tag{51}$$

Then the homogenized problem becomes:

$$\min_{\bar{\mathbf{w}}} \bar{\mathbf{w}}^H \mathbf{R} \bar{\mathbf{w}} \quad \text{s.t. } |\bar{w}_i|^2 = 1 \ \forall i \tag{52}$$

This is a unit-modulus quadratic programming (UQP) problem. Now this can be solved via ADMM. —

Step 1: ADMM Formulation Eqn. 52 can be rewritten as:

$$\min_{\bar{\mathbf{x}}, \bar{\mathbf{w}}} \frac{1}{2} \bar{\mathbf{x}}^H \mathbf{R} \bar{\mathbf{x}} + \tilde{\mathcal{I}}_{\mathcal{F}}(\bar{\mathbf{w}}) \quad \text{s.t. } \bar{\mathbf{x}} = \bar{\mathbf{w}} \tag{53}$$

where $\tilde{\mathcal{I}}_{\mathcal{F}}(\bar{\mathbf{w}})$ is the indicator function for unit modulus constraint:

$$\tilde{\mathcal{I}}_{\mathcal{F}}(\bar{\mathbf{w}}) = \begin{cases} 0 & \text{if } |\bar{w}_i| = 1 \\ \infty & \text{otherwise} \end{cases} \tag{54}$$

—

Step 2: Augmented Lagrangian

$$\mathcal{L}_\mu(\bar{\mathbf{x}}, \bar{\mathbf{w}}, \mathbf{t}) = \frac{1}{2}\bar{\mathbf{x}}^H \mathbf{R}\bar{\mathbf{x}} + \tilde{\mathcal{I}}_{\mathcal{F}}(\bar{\mathbf{w}}) + \mathbf{t}^H(\bar{\mathbf{x}} - \bar{\mathbf{w}}) + \frac{\mu}{2}\|\bar{\mathbf{x}} - \bar{\mathbf{w}}\|_2^2 \tag{55}$$

—

Step 3: ADMM Iterations

$$\bar{\mathbf{x}}^{(k+1)} = \arg\min_{\bar{\mathbf{x}}} \mathcal{L}_\mu(\bar{\mathbf{x}}, \bar{\mathbf{w}}^{(k)}, \mathbf{t}^{(k)}) \tag{56}$$

$$= (\mathbf{R} + \mu I)^{-1}(\mu\bar{\mathbf{w}}^{(k)} - \mathbf{t}^{(k)}) \tag{57}$$

$$\bar{\mathbf{w}}^{(k+1)} = \arg\min_{\bar{\mathbf{w}}} \mathcal{L}_\mu(\bar{\mathbf{x}}^{(k+1)}, \bar{\mathbf{w}}, \mathbf{t}^{(k)}) \tag{58}$$

$$= \mathcal{P}_{\mathcal{F}}\left(\bar{\mathbf{x}}^{(k+1)} + \frac{1}{\mu}\mathbf{t}^{(k)}\right) \tag{59}$$

$$\mathbf{t}^{(k+1)} = \mathbf{t}^{(k)} + \mu(\bar{\mathbf{x}}^{(k+1)} - \bar{\mathbf{w}}^{(k+1)}) \tag{60}$$

—

After convergence, the first $N_t L_t$ elements of $\bar{\mathbf{w}}$ are taken and reshaped into $\mathbf{F}_R$. The baseband precoder is then computed via:

$$\mathbf{F}_B = \gamma(\mathbf{F}_R^H \mathbf{F}_R)^{-1}\mathbf{F}_R^H \mathbf{F}_{\text{opt}}, \quad \gamma = \sqrt{\frac{KN_s}{\|\mathbf{F}_R \mathbf{F}_B\|_F^2}} \tag{61}$$

Algorithm 3: ADMM-AO for Hybrid Precoding

```
**Input:** $\mathbf{F}_{\text{opt}}$
Initialize $\mathbf{F}_R^{(0)}$ with unit modulus, $\mu$, $t^{(0)} = 0$;
Compute $\mathbf{F}_B^{(0)} = (\mathbf{F}_R^{(0)})^\dagger \mathbf{F}_{\text{opt}}$ and scale;
**repeat**
    Fix $\mathbf{F}_B^{(k)}$, form $R = A^H A$ where $A = \mathbf{F}_B^T \otimes I$;
    Run ADMM: **repeat**
        $\bar{x}^{(i+1)} = (R + \mu I)^{-1}(\mu\bar{w}^{(i)} - t^{(i)})$;
        $\bar{w}^{(i+1)} = \mathcal{P}_{\mathcal{F}}\left(\bar{x}^{(i+1)} + \frac{1}{\mu}t^{(i)}\right)$;
        $t^{(i+1)} = t^{(i)} + \mu(\bar{x}^{(i+1)} - \bar{w}^{(i+1)})$;
    **until** *inner stopping criterion met*;
    Reshape $\bar{w}^{(i+1)}$ to $\mathbf{F}_R^{(k+1)}$;
    Update $\mathbf{F}_B^{(k+1)}$ via LS + scaling;
    $k \leftarrow k + 1$;
**until** *stopping criterion met*;
**Output:** $\mathbf{F}_R, \mathbf{F}_B$
```

4 Conclusion

In this work, a structured methodology was developed to address optimization problems using the ADMM. The formulation and implementation were applied to two primary contexts: sparse signal recovery through LASSO and image reconstruction under noise. The theoretical foundation for ADMM was first established through the classical LASSO problem, and its iterative updates were derived and validated through MATLAB simulations. Additionally, a framework for hybrid precoding design was proposed using the ADMM approach, offering promising directions for millimeter-wave communication systems. Although simulation results for the precoding scenario are reserved for future extension, the groundwork has been laid here. All numerical simulations, including those for varying penalty and regularization parameters, demonstrated the scalability and effectiveness of ADMM in both synthetic and real-data-inspired contexts. Practical insights were also gained regarding parameter tuning and convergence behavior. Future work may focus on extending this approach to distributed ADMM algorithms, robust optimization under imperfect or partial channel state information (CSI), and adaptive penalty selection strategies. Moreover, hardware implementation and application to real-world datasets remain as important follow-up directions.

Acknowledgment

We thank Prof. Waheed U. Bajwa for his guidance and constructive feedback during the development of this project.

References

[1] Z. Xu, G. Taylor, H. Li, M. A. Figueiredo, X. Yuan, and T. Goldstein, “Adaptive consensus admm for distributed optimization,” in *International conference on machine learning.* PMLR, 2017, pp. 3841–3850.

[2] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011. [Online]. Available: https://doi.org/10.1561/2200000016

[3] M. Hong, Z.-Q. Luo, and M. Razaviyayn, “Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 3836–3840.

[4] Y. Wang, W. Yin, and J. Zeng, “Global convergence of admm in nonconvex nonsmooth optimization,” *J. Sci. Comput.*, vol. 78, no. 1, p. 29–63, Jan. 2019. [Online]. Available: https://doi.org/10.1007/s10915-018-0757-z

[5] Y. Zhu, “An augmented admm algorithm with application to the generalized lasso problem,” *Journal of Computational and Graphical Statistics*, vol. 26, no. 1, pp. 195–204, 2017.

[6] M.-W. Un, M. Shao, W.-K. Ma, and P. Ching, “Deep mimo detection using admm unfolding,” in *2019 IEEE Data Science Workshop (DSW).* IEEE, 2019, pp. 333–337.

[7] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayundefinedski, *Minimization methods for non-differentiable functions.* Berlin, Heidelberg: Springer-Verlag, 1985.

[8] G. B. Dantzig and P. Wolfe, “Decomposition principle for linear programs,” *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960. [Online]. Available: https://EconPapers.repec.org/RePEc:inm:oropre:v:8:y:1960:i:1:p:101-111

[9] J. BENDERS, “Partitioning procedures for solving mixed-variables programming problems.” *Numerische Mathematik*, vol. 4, pp. 238–252, 1962/63. [Online]. Available: http://eudml.org/doc/131533

[10] J. N. Tsitsiklis, “Problems in decentralized decision making and computation.” Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 1984, ndltd.org (oai:dspace.mit.edu:1721.1/15254).

[11] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.