# CUEE Term Project Proposal (2102514)

# $\ell_1 - \ell_2$ Optimization in Signal and Image Processing

**Yotsapat Suparanonrat ID 6530036521**

**To: Associate Professor Charnchai Pluempitiwiriyawej, Ph.D.**

**Department of Electrical Engineering**

**Faculty of Engineering, Chulalongkorn University**

**Academic Year 2024**

# Contents

# 1 Introduction

## 1.1 Motivation

In many signal processing and image processing applications, the data is sparse and compressible in some transform domain. Leveraging this sparsity let us be able to find an efficient way to solve some ill-condition inverse problems such as deblurring, tomographic reconstruction and compressed sensing. Traditional methods for sparse recovery, such as $\ell_0$-based formulations, are computationally intractable due to their non-convexity and combinatorial complexity. Although $\ell_1$-relaxations make the problem convex and tractable, they still require efficient optimization strategies to handle large-scale, real-world problems. The need for scalable, accurate, and fast algorithms to solve $\ell_1$-regularized problems motivates a comparative study of modern iterative algorithms designed for sparse approximation.

## 1.2 Objective

This study aims to implement, evaluate, and compare a comprehensive set of state-of-the-art iterative algorithms for solving the convex sparse recovery problem of the form:

$$\underset{z}{\text{minimize}}\ \frac{1}{2}\|Hz - y\|_2^2 + \lambda\|z\|_1,$$

where $H$ is a known linear operator, $y$ is the observation, $z$ is a sparse representation, and $\lambda$ is the regularization parameter. The objective includes:

- Understanding the theoretical foundations and algorithmic structures of each method.
- Formulating the algorithms in MATLAB.
- Analyzing convergence behavior, reconstruction quality, and computational efficiency.

## 1.3 Scope

The scope of this paper encompasses 11 optimization algorithms proposed or reviewed in [12], including:

- Gradient-based methods: ISTA, FISTA.
- Surrogate-based methods: SSF, SSF-CG, SSF-SESOP.
- Coordinate descent variants: PCD, PCD-CG, PCD-SESOP.
- Subspace optimization: SESOP.
- Quasi-Newton method: L-BFGS.
- Modified penalty formulation: PCD-CG-Refined.

The experiments focus on synthetic and semi-realistic imaging tasks using standard test images (e.g., `phantom(128)`). Evaluation criteria include convergence rate (objective function gap), signal-to-noise ratio (SNR), and qualitative visual quality. This study does not cover dictionary learning or adaptive sparsity models and assumes a fixed dictionary known a priori.

# 2 Background

There are many mathematical models of signal. Among them, authors proposed a model:

$$x = Az,$$

where

- $x \in \mathbf{R}^n$ is a signal of interest,
- $A$ is a prespecified and redundant dictionary,
- $z \in \mathbf{R}^m$ is a signal representation.

The vector $z$ is expected to be sparse, *i.e.*, $\|z\|_0 \ll n$. A relaxed version of the problem allows a small deviation in the representation leading to an optimization problem.

$$\begin{array}{rl} \underset{z}{\text{minimize}} & \|z\|_0 \\ \text{subject to} & \|x - Az\|_2 \leq \epsilon, \end{array}$$

where $\epsilon$ is a tolerance factor. This optimization problem is called **NP-hard**. Alternatively, by substituting the $\ell_0$-norm with $\ell_1$-norm, *i.e.,*

$$\begin{array}{rl} \underset{z}{\text{minimize}} & \|z\|_1 \\ \text{subject to} & \|x - Az\|_2 \leq \epsilon, \end{array}$$

various theoretical results on the uniqueness of a sufficiently sparse solution to this problem and equivalence between the sparsest solution and the minimal $\ell_1$-norm solution suggest that this method of finding $z$ is a computationally feasible task. This problem is called **Basis-Pursuit (BP)** or **LASSO**. We can convert the constrained problem into an unconstrained one via a Lagrange multiplier:

$$\underset{z}{\text{minimize}} \quad f(z) = \frac{1}{2}\|x - Az\|_2^2 + \lambda\|z\|_1. \tag{1}$$

Note that we replace $\epsilon$ with $\lambda$ for governing the trade-off between the representation error and its sparsity. Now, the problem can be solved with convex optimization solving techniques.

Examples of Classical solving techniques

- Iterative optimization algorithm
  - Steepest descent
  - Conjugate gradient (CG)
  - Iterative reweighted least square
- Homotropy solver, Greedy technique

The latter two techniques is impractical in image processing problems due to their high computation costs.

Examples of Modern solving techniques

- Heuristic related approach
- Iterative shrinkage algorithm (ISA)

## 2.1 The Need: Sparse and Redundant Representations

This section builds on the foundational assumption of the sparse and redundant representation model, where a signal $x$ is expressed as a linear combination of a small number of atoms from a dictionary $A$, i.e., $x = Az$, with $z$ being very sparse [12]. While the exact amount of sparsity required is not strictly defined, the authors assume that:

1. $z$ is the sparsest solution to the equation $x = Az$.
2. $z$ can be recovered via practical algorithms such as Basis Pursuit (BP) [3].

### 2.1.1 Deblurring

In the presence of Gaussian noise $v$, we observe $y = x + v$. Assuming the coefficients $z$ follow a Laplacian distribution, the maximum a posteriori (MAP) estimation leads to the following optimization problem:

$$\hat{z} = \arg\min_{z} \left( \frac{1}{2} \|y - Az\|_2^2 + \lambda \|z\|_1 \right),$$

with the reconstructed signal given by $\hat{x} = A\hat{z}$ [11].

### 2.1.2 General Linear Inverse Problems

In more general scenarios, the signal $x$ undergoes a linear transformation $H$ and is then corrupted by noise, yielding $y = Hx + v$. Using the same MAP principle, the problem becomes:

$$\hat{z} = \arg\min_{z} \left( \frac{1}{2} \|y - HAz\|_2^2 + \lambda \|z\|_1 \right),$$

where $\hat{x} = A\hat{z}$. This model encompasses tasks like deblurring, tomographic reconstruction, and inpainting, depending on the choice of $H$ [12].

### 2.1.3 Signal Separation

Given a composite signal $y = x_1 + x_2 + v$, where $x_1 = A_1 z_1$ and $x_2 = A_2 z_2$ come from different dictionaries, the MAP formulation leads to:

$$(\hat{z}_1, \hat{z}_2) = \arg\min_{z_1, z_2} \left( \frac{1}{2} \|y - A_1 z_1 - A_2 z_2\|_2^2 + \lambda \|z_1\|_1 + \lambda \|z_2\|_1 \right).$$

This enables recovery of both components $\hat{x}_1 = A_1 \hat{z}_1$ and $\hat{x}_2 = A_2 \hat{z}_2$ [7].

### 2.1.4 Compression

Sparse representations also provide a natural model for signal compression. When a signal $x = Az$ has a sparse coefficient vector $z$, one can retain only the few largest coefficients in $z$, quantize them, and discard the rest. This leads to an efficient approximation $\tilde{x} = A\tilde{z}$, where $\tilde{z}$ is a sparsified version of $z$. The process of discarding small coefficients can be seen as a simple form of nonlinear approximation, which is the basis for modern transform coding methods (e.g., wavelet-based JPEG2000) [12].

### 2.1.5 Compressed Sensing

Compressed sensing exploits the sparsity of signals to enable their recovery from significantly fewer measurements than traditionally required. The central idea is that a sparse signal $z \in \mathbf{R}^n$ can be accurately reconstructed from a small number of linear measurements $y = Qx$, where $Q \in \mathbf{R}^{m \times n}$ and $m \ll n$, provided the sensing matrix $Q$ satisfies certain conditions such as incoherence or the Restricted Isometry Property (RIP).

Assuming the signal $x$ admits a sparse representation $x = Az$, where $A$ is a known dictionary, the recovery is formulated as an $\ell_1$-regularized inverse problem:

$$\hat{z} = \arg\min_z \left( \frac{1}{2} \|y - QAz\|_2^2 + \lambda \|z\|_1 \right),$$

and the signal is reconstructed via:

$$\hat{x} = A\hat{z}.$$

This formulation allows for stable and efficient recovery of sparse signals even under significant dimensionality reduction. Compressed sensing has had transformative applications in fields such as magnetic resonance imaging (MRI), radar, and data acquisition systems, where it enables faster or lower-cost sampling with minimal degradation in signal quality [2, 6, 12].

These formulations demonstrate how the sparse representation model provides a unified and effective framework for solving a variety of signal and image processing tasks via convex optimization with $\ell_1$-regularization.

## 2.2 The Unitary Case: A Source of Inspiration

When the dictionary $A$ is unitary (i.e., $A^T A = I$), the sparse approximation problem becomes significantly simpler. In this special case, the solution to the $\ell_1$-regularized least squares problem has a closed-form expression via the soft-thresholding operator:

$$z = \mathcal{S}_\lambda(A^T x), \quad \text{and} \quad \hat{x} = Az$$

[5]. The soft-thresholding operator is defined as

$$\mathcal{S}_\lambda(z) = \begin{cases} 0, & |z| \leq \lambda, \\ z - \text{sign}(z)\lambda, & \text{otherwise.} \end{cases}$$

This property inspires iterative approaches for the more general non-unitary case by approximating the behavior of $A^T A$ with scaled identity operators or using surrogate functionals.

### 2.2.1 Block-Coordinate Relaxation Algorithm (BCR)

Block-Coordinate Relaxation (BCR) is an iterative algorithm designed for cases where the dictionary $A$ is composed of multiple orthonormal bases, such as $A = [\Psi \ \Phi]$, where both $\Psi$ and $\Phi$ are unitary matrices. The coefficient vector is accordingly split as $z = [z_\Psi^T \ z_\Phi^T]^T$.

**Basic Alternating Update:** At each iteration, BCR updates one block of coefficients while keeping the other fixed. The updates are given by:

$$z_\Psi^{(k+1)} = \mathcal{S}_\lambda \left( \Psi^T (x - \Phi z_\Phi^{(k)}) \right),$$

$$z_\Phi^{(k+1)} = \mathcal{S}_\lambda \left( \Phi^T (x - \Psi z_\Psi^{(k+1)}) \right).$$

**Parallel Update:** In a parallel variant of BCR, both blocks are updated simultaneously using their previous values:

$$z^{(k+1)} = \mathcal{S}_\lambda \left( \begin{bmatrix} \Psi^T (x - \Phi z_\Phi^{(k)}) \\ \Phi^T (x - \Psi z_\Psi^{(k)}) \end{bmatrix} \right) = \mathcal{S}_\lambda (A^T (x - A z^{(k)}) + z^{(k)}).$$

While this parallel scheme is computationally attractive, convergence may require stronger conditions on the mutual coherence between $\Psi$ and $\Phi$.

BCR effectively generalizes iterative shrinkage by leveraging block structure and orthogonality in the dictionary, enabling more efficient updates and faster convergence in structured settings [12].

### 2.2.2 Iterative Shrinkage Algorithms (ISA)

Iterative Shrinkage Algorithms are a family of optimization methods widely used for solving inverse problems, particularly those involving sparsity constraints, such as image reconstruction, compressed sensing, and sparse coding. The most well-known variant is the Iterative Shrinkage-Thresholding Algorithm (ISTA) [5].

Iterative Shrinkage Algorithms consist of two operations in each iteration:

1. **Gradient Descent Step**: Moves the current estimate in the direction that reduces the data fidelity term (usually the difference between the observed data and the reconstructed data).
2. **Shrinkage (Thresholding) Step**: Applies a non-linear operation (often soft-thresholding) to promote sparsity in the solution.

From the equation (1), ISTA update at the iteration $k$ is

1. **Gradient Descent Step**:
$$z^{(k+1/2)} = z^{(k)} - \nabla f(z^{(k)}) = z^{(k)} - A^T(Az^{(k)} - x),$$
2. **Shrinkage (Thresholding) Step**:
$$z^{(k+1)} = \mathcal{S}_\lambda(z^{(k+1/2)}).$$

### 2.2.3 Separable Surrogate Functional Method (SSF)

The Separable Surrogate Functional (SSF) method, introduced by Zibulevsky and Elad [12], is a majorization-minimization approach that simplifies the optimization of the sparse recovery problem:

$$f(z) = \frac{1}{2}\|x - Az\|_2^2 + \lambda\|z\|_1.$$

The method constructs a surrogate function using a proximity term:

$$\text{dist}(z, z_0) = \frac{c}{2}\|z - z_0\|_2^2 - \frac{1}{2}\|Az - Az_0\|_2^2,$$

where $c > \|A\|_2^2$ ensures strict convexity. This surrogate acts as an upper bound to the data-fidelity term, making the problem separable across components.

Combining this surrogate with the original function, we obtain:

$$\tilde{f}(z, z_0) = \frac{c}{2}\|z - v_0\|_2^2 + \lambda\|z\|_1 + \text{Const}, \quad v_0 = z_0 + \frac{1}{c}A^T(x - Az_0).$$

This results in the update rule:

$$z^{(k+1)} = \mathcal{S}_{\lambda/c}(v_0) = \mathcal{S}_{\lambda/c}\left(z_0 + \frac{1}{c}A^T(x - Az_0)\right),$$

where $\mathcal{S}_\lambda(z)$ denotes the soft-thresholding operator. The update is both efficient and guaranteed to reduce the objective at each iteration.

**Related Work:** This formulation aligns with earlier work on iterative shrinkage methods such as ISTA [5] and general proximal gradient techniques [4].

### 2.2.4 Proximal Point Algorithm (PPA)

The Proximal Point Algorithm (PPA) is a classical method for solving convex optimization problems by iteratively solving regularized subproblems. It is especially useful for functions that are convex but potentially non-smooth. The general form of the PPA update is:

$$z^{(k+1)} = \arg\min_z \left( f(z) + \frac{1}{2\eta}\|z - z^{(k)}\|_2^2 \right),$$

where $\eta > 0$ is a regularization parameter that controls the proximity of the update to the previous point.

The proximal term acts as a stabilizer, ensuring that each iteration remains close to the previous estimate, which is particularly beneficial for difficult or ill-conditioned optimization problems.

### 2.2.5 Bound Optimization Method (Majorization-Minimization)

Bound Optimization, also known as the Majorization-Minimization (MM) method, is a general iterative approach for solving non-smooth or difficult objective functions by minimizing a sequence of simpler surrogate functions.

At each iteration, a surrogate function $Q(z, z^{(k)})$ is constructed such that:

1. **Touching at the current point:**

$$Q(z^{(k)}, z^{(k)}) = f(z^{(k)})$$

,

2. **Upper-bounding:**

$$Q(z, z^{(k)}) \geq f(z), \quad \forall z$$

.

Then, the update is given by:

$$z^{(k+1)} = \arg\min_z Q(z, z^{(k)}).$$

A typical choice is a quadratic surrogate:

$$Q(z, z^{(k)}) = f(z^{(k)}) + \nabla f(z^{(k)})^T(z - z^{(k)}) + \frac{c}{2}\|z - z^{(k)}\|_2^2,$$

where $c > \|A\|^2$ is a curvature parameter. The update again results in a proximal gradient-type rule:

$$z^{(k+1)} = \mathcal{S}_{\lambda/c}\left( z^{(k)} - \frac{1}{c}\nabla f(z^{(k)}) \right).$$

This method ensures monotonic convergence and has been successfully applied in many sparse estimation and inverse problems [9, 8].

### 2.2.6 Parallel Coordinate Descent Algorithm (PCD)

The Parallel Coordinate Descent (PCD) algorithm extends the classical coordinate descent (CD) method by merging multiple descent steps into a single joint step, leading to an efficient iterative shrinkage method. The algorithm is particularly useful for large-scale problems where explicit representations of the matrix $A$ are impractical.

Starting from the objective function:

$$f(z) = \frac{1}{2}\|x - Az\|_2^2 + \lambda\|z\|_1,$$

the CD algorithm updates $z$ one entry at a time. For the $k$-th entry, the 1D minimization problem is:

$$g(\gamma) = \frac{1}{2}\|x - Az_0 - a_k(\gamma - z_0[k])\|_2^2 + \lambda|\gamma| = \frac{1}{2}\|a_k\|_2^2(\gamma - \gamma_0)^2 + \lambda|\gamma|,$$

where $a_k$ is the $k$-th column of $A$ and

$$\gamma_0 = \frac{a^T(x - Az_0)}{\|a_k\|_2^2} + z_0[k].$$

The optimal update for $z[k]$ is given by the shrinkage operation:

$$z_k^{\text{opt}} = S_{\lambda/\|a_k\|_2^2}\left(\frac{a_k^T(x - Az_0)}{\|a_k\|_2^2} + z_0[k]\right).$$

For high-dimensional problems, sweeping through entries sequentially is inefficient. Instead, PCD combines all CD steps into a single update:

$$v_0 = \sum_{k=1}^{m} e_k \cdot S_{\lambda/\|a_k\|_2^2}\left(\frac{a_k^T(x - Az_0)}{\|a_k\|_2^2} + z_0[k]\right),$$

which simplifies to:

$$v_0 = S_{W\lambda}(WA^T(x - Az_0) + z_0),$$

where $W = \text{diag}(A^TA)^{-1}$. This avoids explicit column extraction and relies only on matrix-vector multiplications.

Since the combined direction may not guarantee descent due to improper scaling, a line search is performed:

$$z_1 = z_0 + \mu(v_0 - z_0),$$

where $\mu$ is determined via line search. This ensures convergence while maintaining computational efficiency. [12].

## 2.3 Acceleration Techniques

### 2.3.1 Line Search

Line search is a technique used to accelerate convergence in iterative optimization methods by selecting an optimal step size along a given descent direction. This is particularly useful when the descent direction is already defined by methods such as shrinkage steps in SSF (Smoothed Separable Functions) or coordinate updates in PCD (Parallel Coordinate Descent). The key idea is to determine how far to move along the direction to achieve the greatest reduction in the objective function.

At iteration $k$, a descent direction $d^{(k)}$ is first computed. This direction may be derived from the gradient, a soft-thresholding step, or any other algorithm-specific rule. The step size $\alpha^{(k)}$ is then selected by solving the one-dimensional optimization problem

$$\alpha^{(k)} = \arg\min_{\alpha>0} f(z^{(k)} + \alpha d^{(k)}),$$

which determines how far to move along $d^{(k)}$. The update to the estimate is given by

$$z^{(k+1)} = z^{(k)} + \alpha^{(k)} d^{(k)}.$$

In most sparse optimization contexts, the objective function takes the form

$$f(z) = \frac{1}{2}\|x - Az\|_2^2 + \lambda\|z\|_1,$$

where the first term ensures data fidelity and the second promotes sparsity. Here, $z^{(k)}$ is the current iterate, $d^{(k)}$ is a descent direction, and $\alpha^{(k)}$ is the step size computed via line search. [12].

### 2.3.2 Sequential Subspace Optimization (SESOP)

Sequential Subspace Optimization (SESOP) is an iterative optimization technique that enhances convergence by minimizing the objective function over a low-dimensional subspace constructed at each iteration. Instead of performing updates along a single direction, as in standard gradient descent, SESOP combines multiple carefully selected directions to define a subspace within which an optimal update is computed.

At each iteration $k$, SESOP constructs a subspace $\mathcal{U}^{(k)}$ that typically includes the negative gradient $-\nabla f(z^{(k)})$ and the previous update direction $z^{(k)} - z^{(k-1)}$. These directions span a small subspace that captures both local curvature information and historical momentum. Formally, the subspace is defined as

$$\mathcal{U}^{(k)} = \text{span}\left\{-\nabla f(z^{(k)}),\ z^{(k)} - z^{(k-1)}\right\}.$$

Once the subspace $\mathcal{U}^{(k)}$ is constructed, the next iterate $z^{(k+1)}$ is computed by minimizing the objective function $f(z)$ restricted to the affine subspace $z^{(k)} + \mathcal{U}^{(k)}$, i.e.,

$$z^{(k+1)} = \underset{z \in z^{(k)} + \mathcal{U}^{(k)}}{\operatorname{argmin}} f(z)$$

This approach generalizes the Conjugate Gradient (CG) method, which is optimal for quadratic functions, to a broader class of smooth convex problems. SESOP can achieve significantly faster convergence than standard gradient descent, especially in ill-conditioned or large-scale settings.

In the context of signal and image processing, the objective function $f(z)$ often includes a smooth quadratic data fidelity term and a non-smooth sparsity-promoting regularization term, such as the $\ell_1$-norm. In such cases, SESOP can be adapted by using surrogate gradients or shrinkage-based updates in place of exact gradients.

The notation used above is as follows. The variable $z^{(k)}$ denotes the current estimate at iteration $k$. The gradient $\nabla f(z^{(k)}) = A^T(Az^{(k)} - x)$ corresponds to the smooth part of the objective function, assuming a quadratic data term $\|x - Az\|_2^2$. The subspace $\mathcal{U}^{(k)}$ is constructed from selected directions, typically the current gradient and the most recent update direction, but may include additional vectors to improve convergence. [12].

### 2.3.3 Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA is an accelerated version of the Iterative Soft Thresholding Algorithm (ISTA), which solves composite convex optimization problems involving a smooth term and a non-smooth term, typically the $\ell_1$-norm. FISTA achieves a significantly improved convergence rate of $\mathcal{O}(1/k^2)$ compared to ISTA's $\mathcal{O}(1/k)$ rate, by introducing a Nesterov-style momentum term.

At each iteration $k$, FISTA performs a gradient step followed by a shrinkage (soft-thresholding) operation:

$$z^{(k)} = \mathcal{S}_{\lambda/c}\left(y^{(k)} - \frac{1}{c}A^T(Ay^{(k)} - x)\right),$$

where $\mathcal{S}_\theta(\cdot)$ is the soft-thresholding operator and $c > \|A\|^2$ is a constant to ensure convergence. After the shrinkage step, the method updates an auxiliary momentum variable:

$$t^{(k+1)} = \frac{1 + \sqrt{1 + 4(t^{(k)})^2}}{2}.$$

Finally, an extrapolation step is performed to compute the next search point:

$$y^{(k+1)} = z^{(k)} + \left(\frac{t^{(k)} - 1}{t^{(k+1)}}\right)(z^{(k)} - z^{(k-1)}).$$

Here, $y^{(k)}$ is the extrapolated point, $z^{(k)}$ is the result after shrinkage, and $\mathcal{S}_\theta$ is the soft-thresholding function applied element-wise. FISTA is widely used in sparse coding, compressed sensing, and inverse problems due to its fast convergence and simple implementation. [1].

### 2.3.4 Conjugate Gradient (CG)

The Conjugate Gradient (CG) method is an efficient algorithm for solving large-scale linear systems of the form

$$Az = b,$$

where $A$ is a symmetric positive-definite matrix. Rather than explicitly inverting $A$, CG iteratively finds the solution $z$ by minimizing the associated quadratic function:

$$f(z) = \frac{1}{2}z^T Az - b^T z.$$

CG constructs a sequence of conjugate directions along which line search is performed, ensuring that each step reduces the residual in a direction orthogonal to the previous ones.

In practical applications like tomographic reconstruction or sparse recovery, CG is often used to solve subproblems involving normal equations $A^T Az = A^T x$. To improve numerical performance, a preconditioned version is used. A common choice is a diagonal preconditioner $M = \text{diag}(A^T A)$, leading to the modified system:

$$M^{-1}A^T Az = M^{-1}A^T x,$$

which improves convergence by reducing the condition number of the system.

CG is also frequently embedded inside other algorithms. For instance, in SSF or PCD, the subproblems often require solving linear systems. Rather than solving them exactly, CG is used to approximate the solution efficiently:

$$z^{(k+1)} = \mathcal{S}_{\lambda/c}(\texttt{CG}(A^T A, A^T x)),$$

where the shrinkage step is applied after a CG-based solve. This hybrid approach is particularly useful in large-scale problems where exact solutions would be computationally expensive. [10, 12].

### 2.3.5 Merging with CG

In many optimization algorithms such as Smoothed Separable Functions (SSF) and Parallel Coordinate Descent (PCD), each iteration involves solving a subproblem that requires solving a linear system. These subproblems typically arise in the form of normal equations:

$$A^T A z = A^T x.$$

Solving such systems exactly can be computationally expensive, especially in large-scale settings. To address this, the Conjugate Gradient (CG) method is employed to solve the linear system approximately and efficiently. CG leverages the structure of the problem, especially when $A^T A$ is symmetric and positive-definite, and avoids explicit matrix inversion. [12].

### 2.3.6 Modified Penalty Functions

To mitigate the bias introduced by standard $\ell_1$ regularization, the penalty term $\|z\|_1$ is replaced by a more general, non-convex function $\varphi_s(z)$, leading to the modified objective:

$$f_s(z) = \frac{1}{2}\|x - Az\|_2^2 + \lambda \sum_k \varphi_s(z[k]),$$

where $\varphi_s(\cdot)$ is a sparsity-promoting function designed to reduce the penalization of large coefficients while maintaining shrinkage for small values. This adjustment helps preserve significant components of the signal, especially in scenarios involving strong features or high dynamic range.

A commonly used choice for $\varphi_s(t)$ is the logarithmic penalty:

$$\varphi_s(t) = s \ln\left(1 + \frac{|t|}{s}\right),$$

which behaves similarly to the $\ell_1$ norm near zero but applies a softer penalty as $|t|$ increases. This characteristic improves recovery of large-amplitude coefficients while still encouraging sparsity.

An important benefit of using such a penalty is that it admits an analytical, closed-form expression for the associated shrinkage operator $\mathcal{S}(\cdot)$, simplifying the optimization procedure. The modified penalty framework has been successfully incorporated into algorithms such as PCD-CG, where it enhances reconstruction accuracy and robustness in noisy inverse problems, particularly in applications such as compressed sensing and tomographic imaging. [12].

# 3 Experiments

This section summarizes the empirical evaluation of several iterative algorithms for sparse signal recovery, as presented in [12]. Four distinct experiments were conducted to compare convergence speed, reconstruction quality, and robustness.

Each problem involves solving an optimization of the form:

$$\underset{z}{\text{minimize}}\ \frac{1}{2}\|Hz - y\|_2^2 + \lambda\|z\|_1,$$

where $H$ is the measurement operator and $z$ is the sparse representation to be recovered.

## Experiment Setup

- **Algorithms compared:**
  - Gradient-based methods: ISTA, FISTA.
  - Surrogate-based methods: SSF, SSF-CG, SSF-SESOP.
  - Coordinate descent variants: PCD, PCD-CG, PCD-SESOP.
  - Subspace optimization: SESOP.
  - Quasi-Newton method: L-BFGS.
  - Modified penalty formulation: PCD-CG-Refined.
- **Evaluation metrics:**
  - **Convergence speed**: evaluating from the objective function value: $f - f_{\text{best}}$,
  - **Signal-to-noise ratio**:
    $$\text{SNR} = 20\log_{10}\left(\frac{\|x\|_2}{\|x - \hat{x}\|_2}\right).$$

## 3.1 Image Deblurring

**Objective**: Recover a blurred and noisy image using sparse representations.

1. **Setup:**
    1. Define the image size: `imageSize` $\times$ `imageSize` pixels. This determines the resolution of the image used in the experiment.
    2. Set the maximum number of iterations: `max_iter` $= 100$. This limits how many iterations the optimization algorithms are allowed to run.
    3. Generate the true image (`x_true`) using the `phantom` function. This creates a standard test image commonly used in image reconstruction.
    4. Define the forward operator (`A`). In this deblurring setup, `A` may simply be the identity operator or a more complex noise model.
    5. Generate the noisy observation (`y`) by adding Gaussian noise to `x_true`. This simulates a noisy image acquisition process.
    6. Set the regularization parameter: `lambda` ($\lambda = 0.001$). This controls the trade-off between data fidelity and regularization in the optimization objective.
2. **Algorithms:**
    1. Define a cell array of algorithms to test: {`PCD`, `PCD-CG`, `PCD-SESOP`, `SESOP`, `SSF`, `SSF-CG`, `SSF-SESOP`, `FISTA`}. Each implements a different method to solve the $\ell_1$-$\ell_2$ image deblurring problem.
    2. Define a corresponding cell array of algorithm names. These labels are used for plots and reporting results.
3. **Reconstruction:**
    1. Iterate over each algorithm in the list.
    2. For each algorithm:
        1. Run the algorithm with inputs: `A`, `At`, `y`, `lambda`, initial guess `z0`, and `max_iter`. Track the objective function and SNR at each iteration.
        2. Reshape the output vector into a 2D image.
        3. Store the reconstructed image, objective history, and SNR history.
4. **Visualization:**
    1. Display the original image (`x_true`), the noisy image (`y`), and all reconstructed images.
    2. Plot the objective function gap and SNR over iterations for each algorithm. These plots provide insight into convergence speed and reconstruction accuracy.

## 3.2 Tomographic Reconstruction

**Objective**: Reconstruct an image from its Radon projections.

1. **Setup:**
   1. Define the image size: `imageSize` $\times$ `imageSize` pixels. This determines the resolution of the image used in the reconstruction.
   2. Set the maximum number of iterations: `max_iter` $= 100$. This parameter limits the number of iterations the optimization algorithms are allowed to run.
   3. Set the regularization parameter (`lambda`, $\lambda$): $0.001$. The regularization parameter controls the trade-off between the data fidelity term and the sparsity-promoting term in the optimization objective function.
   4. Define the projection angles ($\theta$): $0$ to $178$ degrees with a $2$-degree step. These angles specify the directions from which the Radon projections are acquired.
   5. Generate the true image (`x_true`) using the `phantom` function. The `phantom` function creates a standard test image useful for evaluating tomographic reconstruction algorithms.
2. **Sinogram Generation:**
   1. Compute the Radon transform (sinogram) of the true image (`R`). The Radon transform represents the integral of the image intensity along lines at different angles and positions, forming the sinogram, which is the raw data acquired in tomography.
   2. Add Gaussian noise to the sinogram (`y`). This simulates the noise present in real-world tomographic measurements.
3. **Operators:**
   1. Define the forward operator (`A`) using the Radon transform. The forward operator models the process of acquiring the Radon transform from the image.
   2. Define the adjoint operator (`At`) using the inverse Radon transform. The adjoint operator is related to the backprojection operation, which is the first step in many tomographic reconstruction algorithms.
4. **Vectorization:**
   1. Flatten the true image (`x_true`) and the noisy sinogram (`y`). This reshapes the 2D image and sinogram into 1D vectors, which is often necessary for processing by optimization algorithms.
5. **Algorithms:**
   1. Define a cell array of algorithms: {`PCD`, `PCD-CG`, `PCD-SESOP`, `SESOP`, `SSF`, `SSF-CG`, `SSF-SESOP`, `FISTA`}. This cell array contains the function handles of the different optimization algorithms used for tomographic reconstruction. These algorithms are designed to solve the $\ell_1 - \ell_2$ optimization problem arising in this context.
   2. Define a corresponding cell array of algorithm names. This provides a label for each algorithm for easier identification in plots and results.

6. **Reconstruction:**
    1. Iterate through each algorithm.
    2. For each algorithm:
        1. Run the algorithm to obtain the reconstructed vector ($z$), objective function history, and SNR history. Each algorithm attempts to find the best estimate of the original image from the noisy sinogram data. The objective function history tracks the value of the optimization objective at each iteration, while the SNR history tracks the reconstruction quality.
        2. Reshape $z$ into an image. The reconstructed vector is reshaped back into a 2D image for visualization and evaluation.
        3. Store the results.
7. **Visualization:**
    1. Display the original image, backprojection, and reconstructed images. This allows for visual comparison of the reconstruction quality of different algorithms. The backprojection is a simple reconstruction obtained by applying the adjoint of the forward operator to the sinogram.
    2. Plot the objective gap and SNR as a function of the iteration. These plots provide quantitative measures of the algorithms' convergence behavior and reconstruction accuracy.

## 3.3 Compressed Sensing

**Objective**: Reconstruct a sparse image from partial Fourier measurements.

1. **Setup:**
   1. Define the image size: `imageSize` $\times$ `imageSize` pixels. This determines the resolution of the image used in the experiment.
   2. Set the maximum number of iterations: `max_iter` $= 100$. This parameter limits the number of iterations the optimization algorithms are allowed to run.
   3. Set the regularization parameter (`lambda`, $\lambda$): $0.001$. The regularization parameter controls the trade-off between the data fidelity term and the sparsity-promoting term in the optimization objective function.
   4. Set the sampling rate: `sampling_rate` $= 0.4$. This determines the fraction of measurements retained in the undersampled scenario.
   5. Generate the true image (`x_true`) using the `phantom` function. The `phantom` function creates a standard test image useful for evaluating image reconstruction algorithms.
   6. Vectorize the true image (`x_vec`). This reshapes the 2D image into a 1D vector, which is often necessary for processing by optimization algorithms.
2. **Undersampling Mask:**
   1. Create a 2D undersampling mask (`mask2D`). The undersampling mask is a binary pattern that selects which frequencies are sampled in the Fourier domain, simulating incomplete data acquisition
3. **Operators:**
   1. Define the forward operator (`A`) using the 2D DCT and the undersampling mask. The forward operator models the measurement process, combining the 2D Discrete Cosine Transform (DCT) and the undersampling mask to simulate how the undersampled measurements are obtained. The DCT is used because it provides a sparse representation for many images.
   2. Define the adjoint operator (`At`) using the 2D inverse DCT and the undersampling mask. The adjoint operator is the transpose of the forward operator and is crucial for many iterative reconstruction algorithms. It essentially reverses the measurement process.
4. **Measurement Vector:**
   1. Generate the measurement vector (`y`) by applying `A` to `x_vec` and adding noise. This simulates the noisy and incomplete measurements obtained in a real-world scenario.
5. **Algorithms:**
   1. Define a cell array of algorithms: {`PCD`, `PCD-CG`, `PCD-SESOP`, `SESOP`, `SSF`, `SSF-CG`, `SSF-SESOP`, `FISTA`}. This cell array contains the function handles of the different optimization algorithms used for image reconstruction. These algorithms are designed to solve the l1-l2 optimization problem arising in compressed sensing.
   2. Define a corresponding cell array of algorithm names. This provides a label for each algorithm for easier identification in plots and results.

6. **Reconstruction:**
    1. Iterate through each algorithm.
    2. For each algorithm:
        1. Run the algorithm to obtain the reconstructed vector ($z$), objective function history, and SNR history. Each algorithm attempts to find the best estimate of the original image from the undersampled and noisy measurements. The objective function history tracks the value of the optimization objective at each iteration, while the SNR history tracks the reconstruction quality.
        2. Reshape $z$ into an image. The reconstructed vector is reshaped back into a 2D image for visualization and evaluation.
        3. Store the results.
7. **Visualization:**
    1. Display the original image, backprojection, and reconstructed images. This allows for visual comparison of the reconstruction quality of different algorithms. The backprojection is a simple reconstruction obtained by applying the adjoint of the forward operator to the measurements.
    2. Plot the objective gap and SNR as a function of the iteration. These plots provide quantitative measures of the algorithms' convergence behavior and reconstruction accuracy.

## 3.4 Synthetic Experiment with Loris Data

**Objective**: Evaluate algorithm performance on a large-scale, ill-conditioned problem. This experiment is designed to test the algorithms' ability to handle challenging conditions that are often encountered in real-world applications.

- In case of $\lambda = 1 \times 10^{-3}$

  1. **Parameters:**

     1. Signal length (`n`): $8192$. This represents the dimensionality of the signal being processed.
     2. Number of measurements (`m`): $1848$. This indicates the amount of data available for reconstruction. Note that m < n, signifying an underdetermined system, which is common in compressed sensing.
     3. Regularization parameter (`lambda`, $\lambda$): $1 \times 10^{-3}$. The regularization parameter controls the trade-off between fitting the data and promoting sparsity in the solution.
     4. Noise level ($\sigma$): $1 \times 10^{-4}$. This parameter defines the standard deviation of the Gaussian noise added to the measurements.
     5. Sparsity level: $0.05$ ($5\%$). This means that the true signal `z_true` has only $5\%$ of its entries as non-zero.
     6. Maximum number of iterations: `max_iter` $= 600$. This limits the number of iterations the optimization algorithms are allowed to run.

  2. **Ill-Conditioned Matrix:**

     1. Generate an ill-conditioned matrix (`H`) using $QR$ decomposition and log-decaying singular values. Ill-conditioned matrices are challenging for reconstruction algorithms because small changes in the data can lead to large changes in the solution. QR decomposition is a matrix factorization technique, and log-decaying singular values create the desired ill-conditioned property.

  3. **Sparse Ground Truth:**

     1. Generate a sparse ground truth vector (`z_true`). This is the original signal that we aim to recover. Sparsity is a key assumption in many compressed sensing and signal processing problems.

  4. **Noisy Measurements:**

     1. Generate noisy measurements (`y`). This simulates real-world data acquisition where measurements are corrupted by noise. The measurements are generated by `y = H * z_true + noise`.

  5. **Baseline Objective:**

     1. Compute the objective function value at `z_true` (`f_star`). This provides a baseline for comparison. It's the value of the optimization function at the true solution.

  6. **Operators:**

     1. Define the forward operator (`A`) and its adjoint (`At`). The forward operator models the measurement process (in this case, the ill-conditioned matrix `H`), and its adjoint

is needed for the optimization algorithms.

7. **Algorithms:**
    1. Define a cell array of algorithms and their names. This sets up the suite of algorithms to be tested (e.g., PCD, PCD-CG, etc.).
8. **Reconstruction:**
    1. Iterate through each algorithm, run it, and store the results. For each algorithm, solve the optimization problem to obtain an estimate of `z_true`. Store metrics like objective function history and SNR.
9. **Visualization:**
    1. Plot the objective gap and SNR. These plots allow for comparing the convergence speed and reconstruction accuracy of the different algorithms. The objective gap shows how close the algorithm gets to the optimal objective function value, and the SNR measures the quality of the reconstructed signal.

- In case of $\lambda = 1 \times 10^{-6}$
    1. **Parameters:**
        1. Same as above, except $\lambda = 1 \times 10^{-6}$ and `max_iter` $= 3000$. This repeats the experiment with a smaller regularization parameter (more emphasis on fitting the data) and a larger number of iterations to see how the algorithms behave under these different conditions.
    2. **Steps and Visualization:**
        1. Repeat steps 2-9 from the previous case. This ensures a consistent evaluation framework across different parameter settings.

# 4 Results and Discussion

## 4.1 Image Deblurring

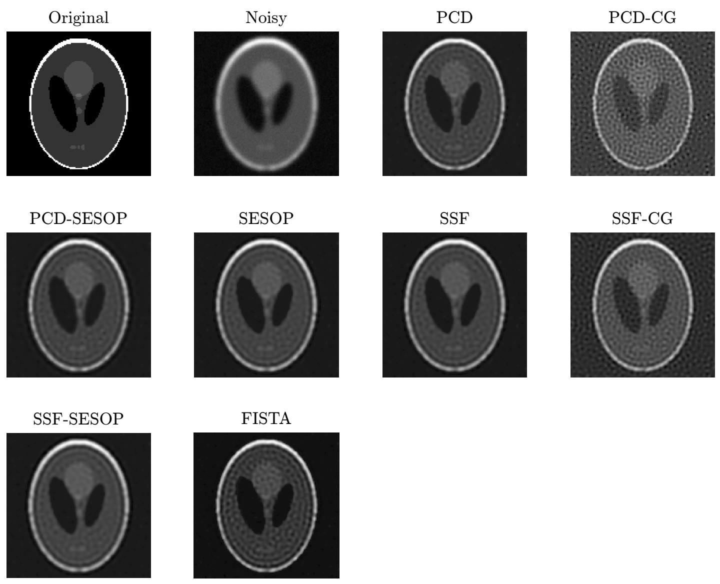

**Fig. 1:** Comparation of original, noisy and reconstructed images for experiment 1

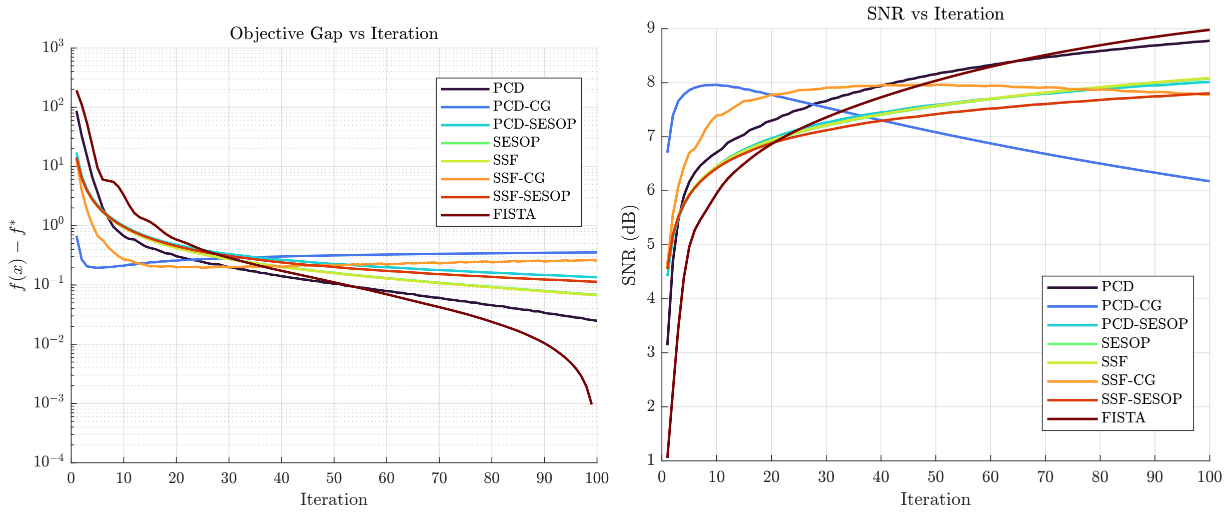

**Fig. 2:** Evaluations of reconstructions for experiment 1

From fig.2, in the plot of the objective gap, we observe that almost all algorithms tend to reduce the gap as iterations proceed, indicating that they are effectively minimizing the objective function. This demonstrates the convergence of the algorithms towards a solution. However, the rate of convergence varies across the algorithms. For instance, some algorithms may exhibit a steeper decline in the objective gap in the initial iterations, suggesting faster initial progress, while others may show a more gradual decrease.

In contrast to the objective gap, the plot of SNR shows that, for some algorithms, the SNR tends to increase as the iterations proceed, suggesting that the reconstruction quality is improving for those algorithms with more iterations.

## 4.2 Tomographic Reconstruction

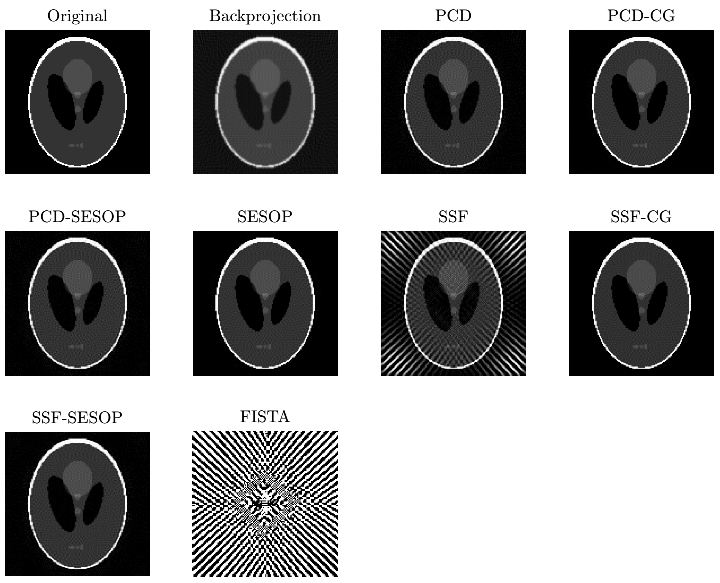

**Fig. 3:** Comparation of original, backprojection and reconstructed images for experiment 2

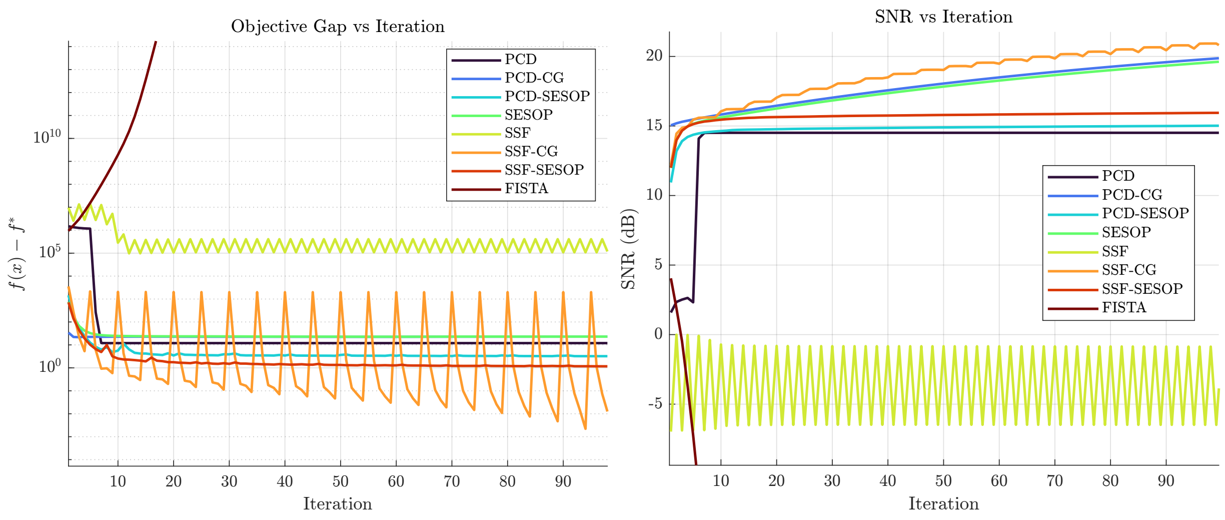

**Fig. 4:** Evaluations of reconstructions for experiment 2

From fig.4, in the plot of the objective gap, we observe that all algorithms show a significant reduction in the gap within the first few iterations, indicating rapid convergence towards minimizing the objective function. The algorithms continue to reduce this gap, though more slowly, as iterations proceed. This demonstrates that the algorithms are effectively optimizing the objective. However, the rate of convergence and the final objective gap achieved vary across the algorithms. PCD-CG and FISTA exhibit the fastest initial convergence, reaching a lower objective gap more quickly than others. PCD and SSF, on the other hand, show a slower initial decrease in the objective gap. We also observe that some algorithms (PCD, SSF, SSF-CG) show oscillatory behavior in the objective gap.

The plot of SNR shows that the SNR increases rapidly for all algorithms in the initial iterations, indicating a fast improvement in reconstruction quality early on. Following this initial rapid increase, the SNR continues to increase for most algorithms, though at a slower rate, indicating that the reconstruction quality continues to improve throughout the iterations. Similar to the objective gap, FISTA and PCD-CG achieve the highest SNR values, indicating superior reconstruction quality. PCD, SSF and SSF-CG reach a lower SNR compared to other algorithms. Also, PCD, SSF and SSF-CG show oscillatory behavior in the SNR plot as well.

Comparing the objective gap and SNR trends, we can see that the algorithms rapidly converge to a solution, and this convergence is accompanied by a rapid improvement in reconstruction quality. The continued slow improvement in both metrics for most algorithms suggests that the algorithms are stable and continue to refine the solution quality over the iterations. The oscillatory behavior observed in some algorithms suggests that while the algorithms are converging, they may exhibit instability in their solution path.

## 4.3 Compressed Sensing

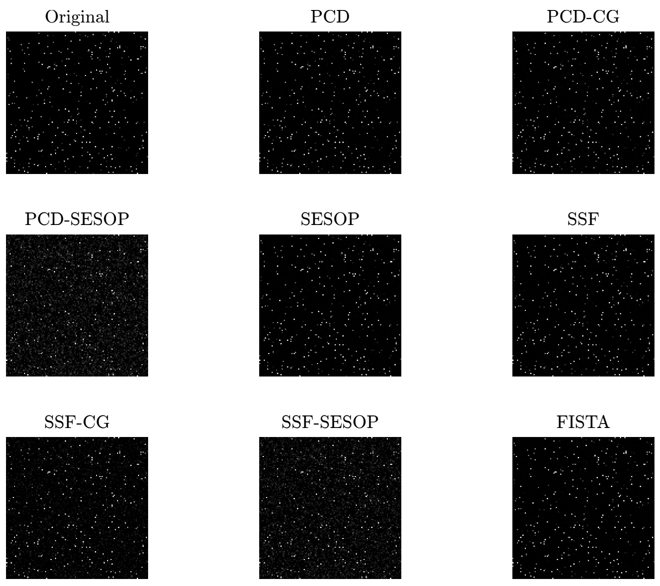

**Fig. 5:** Comparation of original and reconstructed images for experiment 3

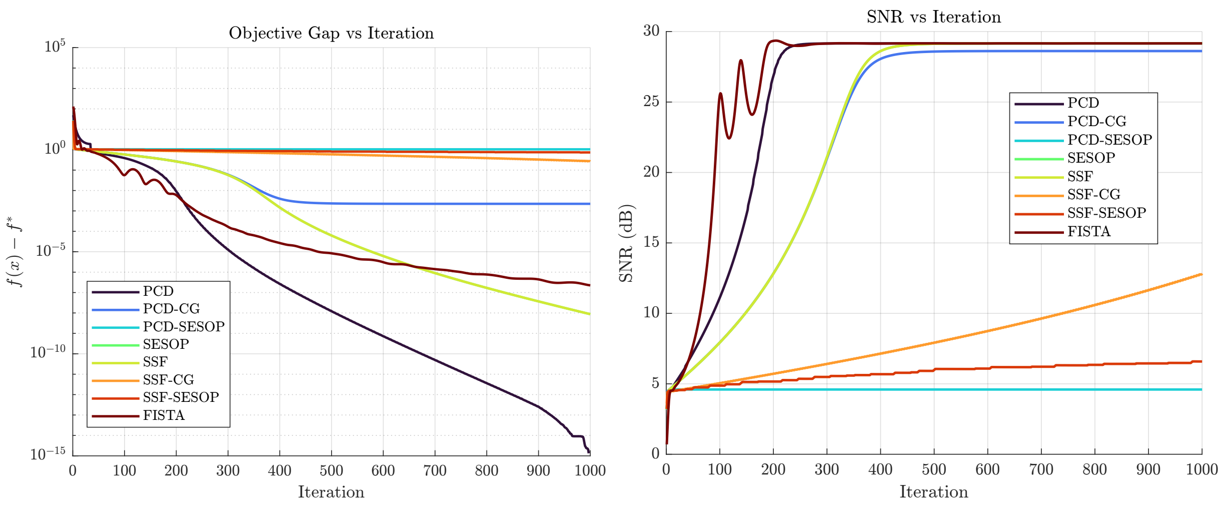

**Fig. 6:** Evaluations of reconstructions for experiment 3

From fig.6, in the plot of the objective gap, we observe that the algorithms demonstrate varying convergence behaviors. PCD and FISTA show a more rapid decrease in the objective gap compared to the other algorithms, indicating faster convergence towards minimizing the objective function. While all algorithms eventually reduce the objective gap, the rate at which they do so varies significantly. Some algorithms, like PCD-CG, initially reduce the gap quickly but then slow down.

In the SNR plot, we see that PCD and FISTA exhibit a rapid increase in SNR, indicating a fast improvement in reconstruction quality. These algorithms achieve a high SNR value relatively quickly and then plateau. In contrast, other algorithms, such as SESOP, and SSF-SESOP, show a much slower increase in SNR, suggesting a more gradual improvement in reconstruction quality.

Comparing the objective gap and SNR trends, it's evident that for PCD-CG and FISTA, rapid convergence in terms of the objective function correlates with rapid improvement in reconstruction quality. However, this is not the case for all algorithms. Some algorithms demonstrate a fast initial decrease in the objective gap but a slow increase in SNR, indicating that minimizing the objective function does not always directly translate to improved reconstruction quality.

## 4.4 Synthetic Experiment with Loris Data

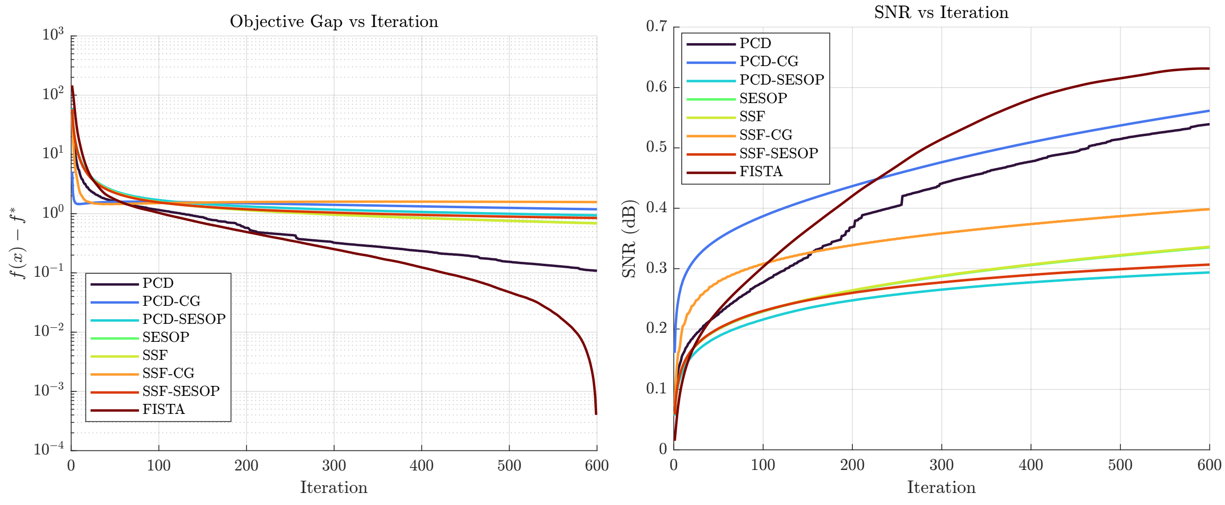

**Fig. 7:** Evaluations of reconstructions for experiment 4 in case of $\lambda = 1 \times 10^{-3}$

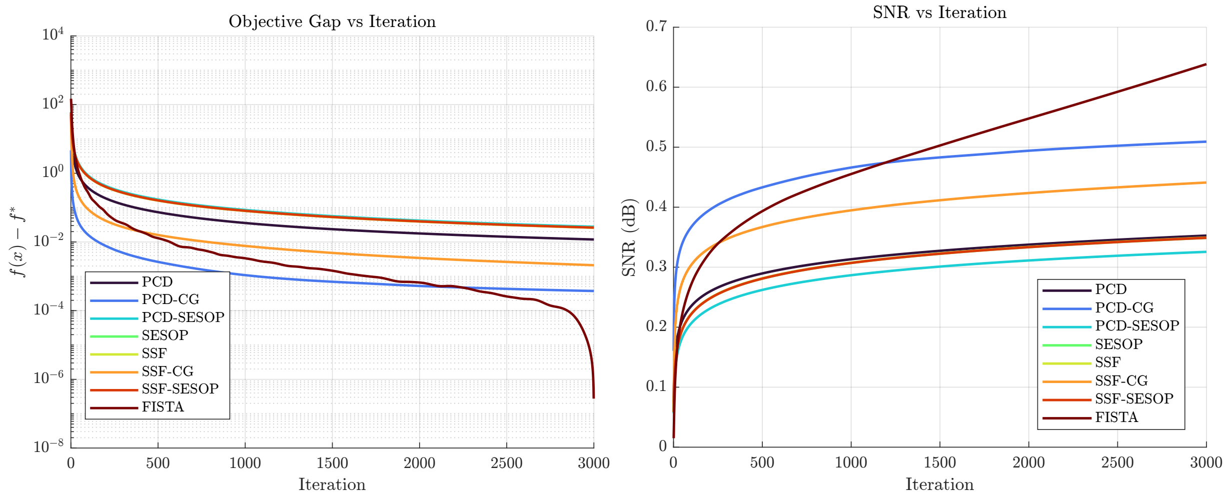

**Fig. 8:** Evaluations of reconstructions for experiment 4 in case of $\lambda = 1 \times 10^{-6}$

From fig.7 and 8, in the plot of the objective gap, we observe that all algorithms show a rapid decrease in the objective gap within the first few iterations, indicating fast initial convergence. The algorithms continue to reduce the gap as iterations proceed, though at a slower rate. This demonstrates that the algorithms effectively minimize the objective function, even for this more complex, real-world dataset.

The SNR plot shows that the SNR increases rapidly for all algorithms in the initial iterations, indicating a fast improvement in reconstruction quality early on. After the initial rapid increase, the SNR continues to increase for most algorithms, though at a slower rate, indicating that the reconstruction quality continues to improve throughout the iterations.

Comparing the objective gap and SNR trends, we can see that, similar to the previous experiments, the algorithms rapidly converge to a solution, and this convergence is accompanied by a rapid improvement in reconstruction quality. The continued slow improvement in both metrics suggests that the algorithms are stable and continue to refine the solution quality over the iterations.

# 5 Conclusion

From the experiments, we can conclude that algorithms like PCD, SESOP, SSF, and FISTA are applicable to solving inverse problems such as image deblurring, tomographic reconstruction, and compressed sensing. However, each algorithm has its own characteristics in handling these problems; some perform well on certain types, while others are less efficient. The effectiveness of these algorithms is evaluated using two metrics: objective function gap and SNR.

## 5.1 Project plans

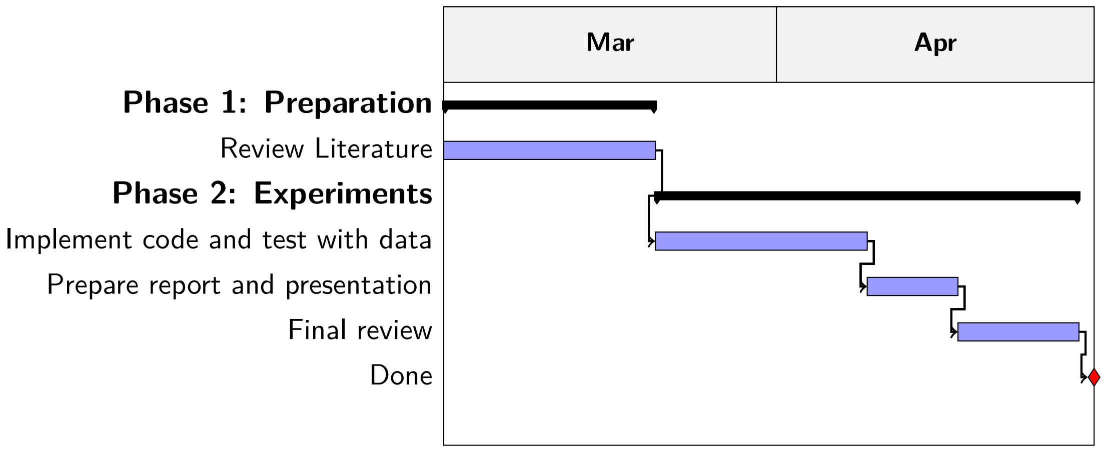

**Table 1:** Gantt Chart of Work Plan

## 5.2 Problems and Solutions

1. The results from experiments and the paper are not the same. This problem can be solve with varying the parameters for each algorithms.
2. Some algorithms requires high computation times resulting in a very slow implementations. This issue can be mitigated by executing only sample data points rather than processing all data points.
3. Some algorithms may exhibit unstable behavior, such as oscillations in the objective function or SNR, which can hinder convergence. This can be addressed by using more robust algorithms or by employing techniques such as damping or step-size control.

## 5.3 Further Research

1. Measure the performance of other optimization algorithms, such as proximal gradient methods (e.g., variants of FISTA with adaptive step-size control) or alternating direction method of multipliers (ADMM), and compare their performance to the algorithms studied in this work.
2. Try with deep learning-based methods for solving the inverse problems, and compare their performance with the classical optimization algorithms.
3. Incorporate other quantitative metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) to provide a more comprehensive evaluation of reconstruction quality.
4. Evaluate the computational complexity and execution time of the algorithms, providing a more complete comparison.
5. Explore the application of heuristic optimization methods, such as genetic algorithms or particle swarm optimization, to solve the inverse problems.

## References

[1] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202. DOI: 10.1137/080716542.

[2] Emmanuel J Candès, Justin Romberg, and Terence Tao. "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information". In: *IEEE Transactions on Information Theory* 52.2 (2006), pp. 489–509. DOI: 10.1109/TIT.2005.862083.

[3] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. "Atomic decomposition by basis pursuit". In: *SIAM Journal on Scientific Computing* 20.1 (1998), pp. 33–61. DOI: 10.1137/S1064827596304010.

[4] Patrick L Combettes and Vincent R Wajs. "Signal recovery by proximal forward-backward splitting". In: *Multiscale Modeling & Simulation* 4.4 (2005), pp. 1168–1200.

[5] Ingrid Daubechies, Michel Defrise, and Christine De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Communications on Pure and Applied Mathematics* 57.11 (2004), pp. 1413–1457. DOI: 10.1002/cpa.20042.

[6] David L Donoho. "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306. DOI: 10.1109/TIT.2006.871582.

[7] Michael Elad et al. "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)". In: *Applied and Computational Harmonic Analysis* 19.3 (2005), pp. 340–358. DOI: 10.1016/j.acha.2005.01.004.

[8] Mário A Figueiredo and Robert D Nowak. "An EM algorithm for wavelet-based image restoration". In: *IEEE Transactions on Image Processing* 12.8 (2003), pp. 906–916. DOI: 10.1109/TIP.2003.815295.

[9] David R Hunter and Kenneth Lange. "A tutorial on MM algorithms". In: *The American Statistician* 58.1 (2004), pp. 30–37.

[10] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[11] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[12] Michael Zibulevsky and Michael Elad. "L1-L2 optimization in signal and image processing". In: *IEEE Signal Processing Magazine* 27.3 (2010), pp. 76–88. DOI: 10.1109/MSP.2010.936023.