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

Yotsapat Suparanonrat

April 21, 2025

Overview

Section 1

Introduction

Motivation

- In signal and image processing applications, the underlying data is often sparse or compressible in some transform domain (e.g., wavelet or Fourier).
- This sparsity enables efficient solutions to ill-posed inverse problems such as deblurring, tomographic reconstruction, and compressed sensing.
- Sparse recovery can be formulated using ℓ_0 or ℓ_1-regularization techniques to promote sparsity in the solution.
- However, solving these formulations—especially in large-scale problems—requires efficient and scalable optimization algorithms that balance accuracy and computation time.

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:

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

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

Scope

In this paper, it uses 11 optimization algorithms 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 (`phantom(128)`).

Section 2

Background

Introduction

Authors proposed a mathematical model of signals:

$$x = Az,$$

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.

We expect z to be sparse, *i.e.*, $\|z\|_0 \ll n$.

Basis Pursuit (BP)

A relaxed version of the problem allows a small deviation in the representation leading to an optimization problem:

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

where ϵ is a tolerance factor. Note that we substitute $\|z\|_0$ with $\|z\|_1$ for computational feasibility.

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 ϵ with λ for governing the trade-off between the representation error and its sparsity.

The Need: Sparse and Redundant Representations

Hereafter, we shall assume that

1. z is the sparsest solution of $x = Az$,
2. It is reconstructable via practical algorithms.

deblurring
We observe

$$y = x + v,$$

where x is the original signal, y is its noisy version due to an additive Gaussian noise v. The maximum a posteriori (MAP) estimation leads to a problem

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

with the reconstructed signal given by $\hat{x} = A\hat{z}$

General Inverse Problem

In more general scenarios, x is degraded and is then corrupted by noise:

$$y = Hx + v,$$

where H is a linear degradation operation. The MAP estimation leads to a problem

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

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

- This model can be applied in deblurring, deblurring and tomographic reconstruction.
- If $H = I$, the problem reduces to a deblurring.

Compressed Sensing

A sparse signal $z \in \mathbf{R}^n$ can be recovered from $m \ll n$ measurements:

$$y = Qx,$$

where Q is a sensing matrix.

If the signal x has a sparse representation, the recovery is performed via:

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

followed by reconstruction: $\hat{x} = A\hat{z}$.

The Unitary Case: a Source of Inspiration

If the dictionary matrix A is unitary, *i.e.*, $A^T A = I$, we can arrange the equation (1),

$$f(z) = \frac{1}{2}\|z - z_0\|_2^2 + \lambda\|z\|_1, \tag{2}$$

where $z_0 = A^T x$.

Now, we define a soft-thresholding operator as a solution to the ℓ_1-regularized least square problem and it is equal to the reconstructed signal:

$$\hat{z} = \mathcal{S}_\lambda(z_0) = \begin{cases} 0, & |z_0| \leq \lambda, \\ z_0 - \text{sign}(z_0)\lambda, & \text{otherwise.} \end{cases}$$

Block-Coordinate Relaxation Algorithm (BCR)

In general, A does not need to be unitary. Here, suppose A be built as an union of several unitary matrices, *i.e.*,

$$A = \begin{bmatrix} \Psi & \Phi \end{bmatrix},$$

where Ψ and Φ are unitary matrices.
At each iteration, BCR updates one block while keeping the other fixed.

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

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

We can make two updates done in parallel. Then, the closed-form formula for the update in each iteration is given by

$$z^{(k+1)} = \mathcal{S}_{\lambda}(A^T(x - Az^{(k)}) + z^{(k)}).$$

Iterative Shrinkage Algorithms (ISA)

For any matrices A, we can apply this algorithm to reconstruct the signal. ISA consists of two operation in each iteration:

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**:

$$z^{(k+1)} = \mathcal{S}_\lambda(z^{(k+1/2)}).$$

Usually, Iterative Shrinkage Algorithms refer to **Iterative Shrinkage-Thresholding Algorithms (ISTA)**.

Separable Surrogate Functional Method (SSF)

From the equation (1), we add a surrogate objective term:

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

where c is chosen to make the dist function be strictly convex by choosing $c > \|A\|_2^2$.

We can derive the new objective function into a form

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

Hence, the update rule is

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

Bound Optimization Method

This method is also known as a **majorization-minimization method**. We construct a surrogate function $Q(z, z^{(k)})$ approximating the function $\tilde{f}(z)$ with properties

1. Equality at $z = z_0$:
$$Q(z^{(k)}, z^{(k)}) = f(z^{(k)}),$$
2. Upper bounding the original function:
$$Q(z, z^{(k)}) \geq f(z^{(k)}).$$

In one iteration, the solution is given by

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

which ensure the convergence to the local minimum of $\tilde{f}(z)$.

Parallel Coordinate Descent Algorithm (PCD)

- Simple Coordinate Descent Algorithm (CD) update z one entry in each iteration.
- Assume the current solution is $z^{(k)}$. 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 $z_0[k]$ denotes the k-th entry of z_0, 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 update for $z_0[k]$ is

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

- For a large-scale problem, CD requires high computation cost.

- PCD combines all CD steps into a single update:

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

  where $W = \mathrm{diag}(A^T A)^{-1}$.

- To ensure convergence, we apply a **line search**:

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

  where μ is determined via line search.

Acceleration Techniques

- Line Search
- Sequential Subspace Optimization (SESOP)
- Fast Iterative Soft Thresholding Algorithm (FISTA)

Line Search

- A technique to accelerate convergence by choosing an optimal step size along a given direction.
- Typically used as a refinement step in methods like SSF or PCD.

Algorithm:

1. Compute a descent direction $d^{(k)}$, e.g., shrinkage step.
2. Find optimal step size:
$$\alpha^{(k)} = \arg\min_{\alpha>0} f(z^{(k)} + \alpha d^{(k)})$$
3. Update the estimate:
$$z^{(k+1)} = z^{(k)} + \alpha^{(k)} d^{(k)}$$

Notation:

- $z^{(k)}$: Current iterate.
- $d^{(k)}$: Descent direction.
- $\alpha^{(k)}$: Step size.
- $f(z) = \frac{1}{2}\|x - Az\|_2^2 + \lambda\|z\|_1$: Objective function.

Sequential Subspace Optimization (SESOP)

- SESOP improves convergence by minimizing the objective over a low-dimensional subspace.

Algorithm:

1. Construct subspace:

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

2. Solve reduced optimization:

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

Notation:

- $z^{(k)}$: Current estimate.
- $\nabla f(z^{(k)}) = A^T(Az^{(k)} - x)$: Gradient of the smooth part.
- $\mathcal{U}^{(k)}$: Subspace containing chosen directions, including gradient and previous steps.

Fast Iterative Soft Thresholding Algorithm (FISTA)

- FISTA accelerates ISTA using Nesterov-style momentum.
- Achieves a faster convergence rate: $\mathcal{O}(1/k^2)$.

Algorithm:

1. Compute gradient step:

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

2. Update momentum term:

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

3. Extrapolation step:

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

Conjugate Gradient (CG)

CG is an efficient algorithm for solving large-scale symmetric positive-definite systems:

$$Az = b$$

without explicit matrix inversion. By minimizing the quadratic form:

$$\underset{z}{\text{minimize}}\ f(z) = \frac{1}{2} z^T A z - b^T z.$$

Precondition CG: Let $M = \text{diag}(A^T A)$, we can solve the system via

$$M^{-1} A^T A z = M^{-1} A^T x.$$

This method improves in convergence speed.

Merging with CG

- Algorithms like SSF and PCD involve subproblems requiring linear system solutions.
- Instead of directly solving:

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

  CG is applied to solve this approximately and efficiently.
- The modified update becomes:

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

- In **PCD-CG**, CG is used to efficiently compute per-coordinate updates.
- This allows acceleration without loss in convergence guarantees.
- Particularly useful in large-scale problems like tomographic reconstruction and compressed sensing.

Modified Penalty Functions

- To reduce the bias of ℓ_1 regularization, we replace $\|z\|_1$ with $\varphi_s(z)$:

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

  where $\varphi_s(\cdot)$ is the **general penalty function**.
- $\varphi_s(\cdot)$ is non-convex and sparse promoting.
- It gives an analytical closed-form expression for the shrinkage operator $\mathcal{S}(\cdot)$.
- Example:

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

- Behaves like ℓ_1 near zero, but penalizes large coefficients less.
- Helps preserve significant amplitudes and improves sparsity of recovered signals.
- Incorporated into PCD-CG and shown to enhance performance in noisy inverse problems.

Section 3

Experiments, Results and Discussion

Overview

3 Experiments, Results and Discussion
- Image Deblurring
- Tomographic Reconstruction
- Compressed Sensing
- Synthetic Experiment with Loris Data

Metrics

- Objective function value: $f - f_{\text{best}}$,
- Signal-to-noise ratio:

$$\text{SNR} = 20 \log_{10} \left(\frac{\|x\|_2}{\|x - \hat{x}\|_2} \right).$$

Image Deblurring: Experiment

1. **Setup:**
   - Define image size and maximum iterations
   - Generate true image and Gaussian blur filter
   - Generate noisy observation (blur + noise)
   - Set regularization parameter λ
2. **Operators:**
   - Define forward operator $A(z)$: apply convolution via `imfilter` (reshaped)
   - Define adjoint operator $A^T(z)$: same as forward (symmetric kernel)
3. **Algorithms:**
   - PCD, PCD-CG, PCD-SESOP, SESOP, SSF, SSF-CG, SSF-SESOP, FISTA
4. **Reconstruction:**
   - Loop through algorithms
   - Run each algorithm, reshape output image, store results
5. **Visualization:**
   - Display original, noisy, and reconstructed images
   - Plot objective gap and SNR vs. iteration

Image Deblurring: Results(1)

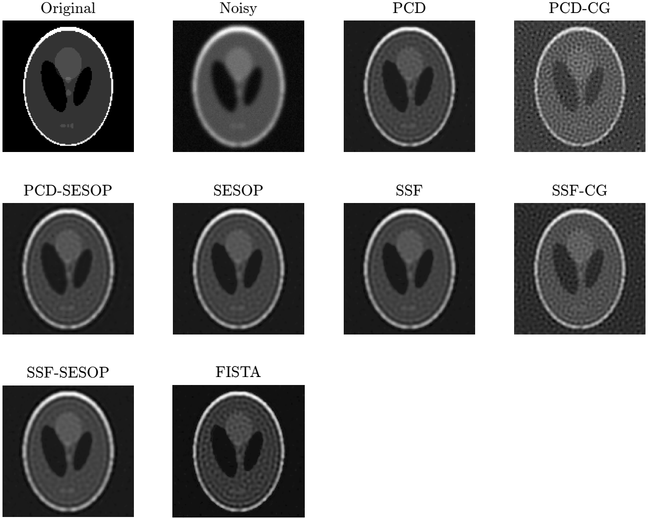

Figure 1: Comparation of original, noisy and reconstructed images for experiment 1

Image Deblurring: Results(2)

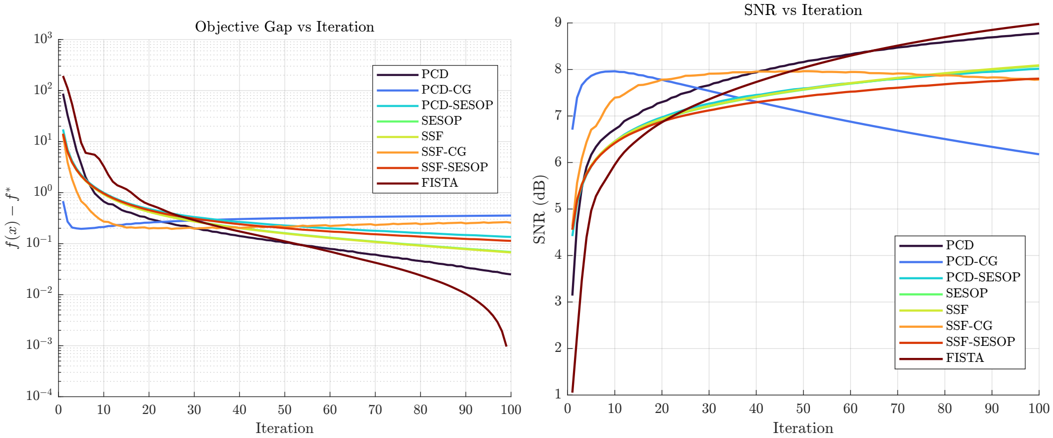

Figure 2: Evaluations of reconstructions for experiment 1

Tomographic Reconstruction: Experiment

1. **Setup:**
   - Define image size and maximum iterations
   - Generate true image using `phantom`, define angles
   - Create Radon sinogram and add Gaussian noise
   - Set regularization parameter λ
2. **Operators:**
   - Forward operator: Radon transform
   - Adjoint operator: inverse Radon transform
3. **Algorithms:**
   - PCD, PCD-CG, PCD-SESOP, SESOP, SSF, SSF-CG, SSF-SESOP, FISTA
4. **Reconstruction:**
   - Loop through algorithms
   - Run each algorithm, reshape output image, store results
5. **Visualization:**
   - Display original, backprojection, and reconstructions
   - Plot objective gap and SNR vs. iteration

Tomographic Reconstruction: Result(1)

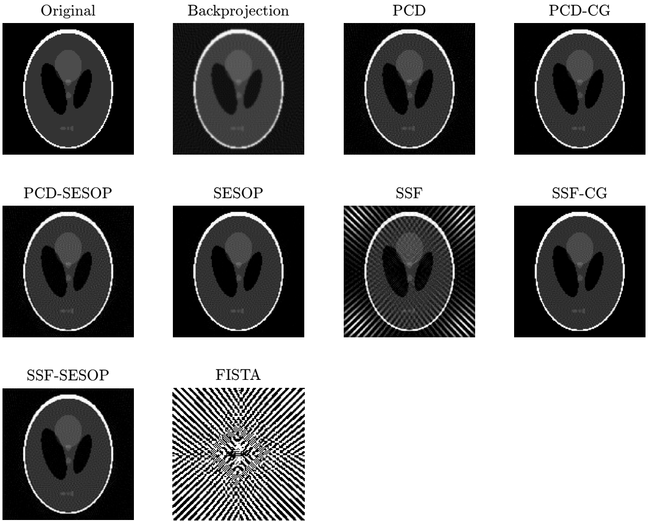

Figure 3: Comparation of original, backprojection and reconstructed images for experiment 2

Tomographic Reconstruction: Result(2)

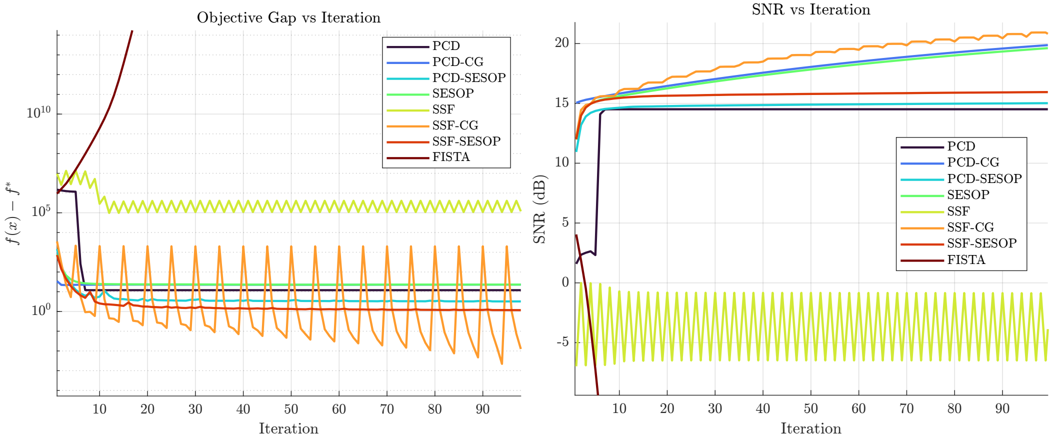

Figure 4: Evaluations of reconstructions for experiment 2

Compressed Sensing: Experiment

1. **Setup:**
   - Set signal size n, measurements m, max iterations
   - Generate sparse ground truth and noisy measurements
   - Create ill-conditioned matrix using QR + decaying singular values
   - Set regularization parameter λ
2. **Operators:**
   - Define forward and adjoint operators
   - Compute reference objective value $f^\star$
3. **Algorithms:**
   - PCD, PCD-CG, PCD-SESOP, SESOP, SSF, SSF-CG, SSF-SESOP, FISTA
4. **Reconstruction:**
   - Run each algorithm and store recovered signal, objective, SNR
5. **Visualization:**
   - Plot objective gap and SNR vs. iteration

Compressed Sensing: Result(1)

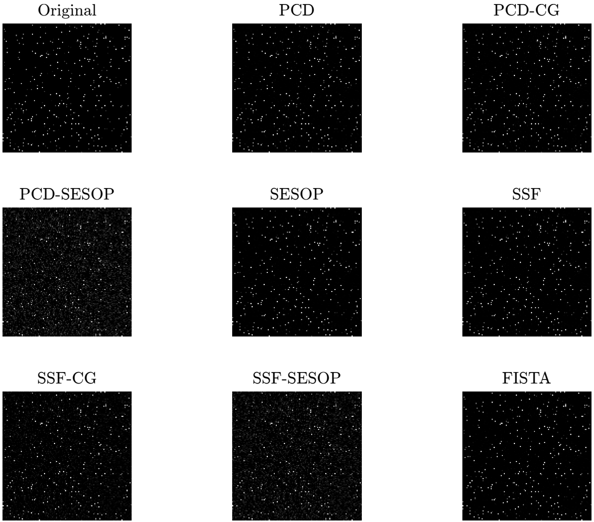

Figure 5: Comparation of original and reconstructed images for experiment 3

Compressed Sensing: Result(2)

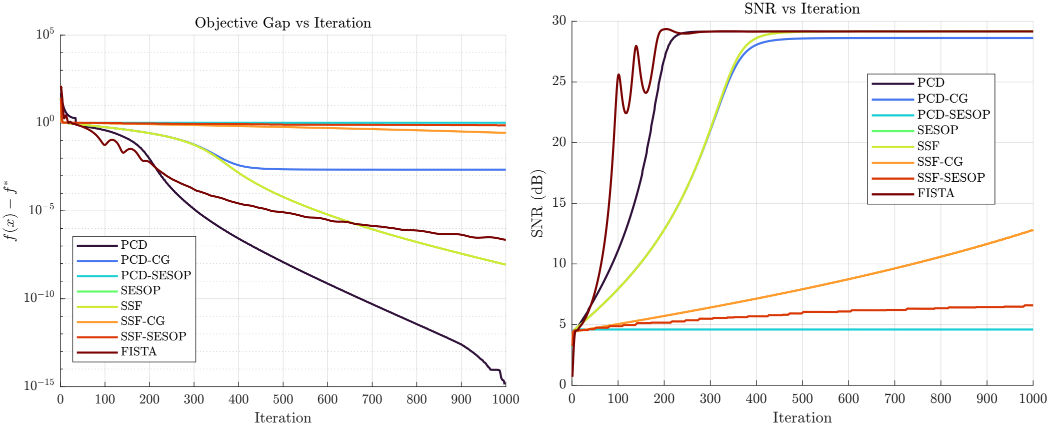

Figure 6: Evaluations of reconstructions for experiment 3

Synthetic Experiment with Loris Data: Experiment

1. **Setup:**
   - Use same n, m, sparsity, and noise level as Exp. 3
   - Use two values of λ: 10^{-3} and 10^{-6}
   - Set `max_iter` $= 600$ (large λ), 3000 (small λ)
   - Generate LORIS sparse ground truth and measurements
2. **Operators:**
   - Define forward and adjoint operators
   - Compute baseline objective value $f^\star$
3. **Algorithms:**
   - PCD, PCD-CG, PCD-SESOP, SESOP, SSF, SSF-CG, SSF-SESOP, FISTA
4. **Reconstruction:**
   - Loop over λ values and algorithms
   - Run, store output signal, SNR, and objective history
5. **Visualization:**
   - Plot results for each λ: objective gap, SNR curves

Synthetic Experiment with Loris Data: Result(1)

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

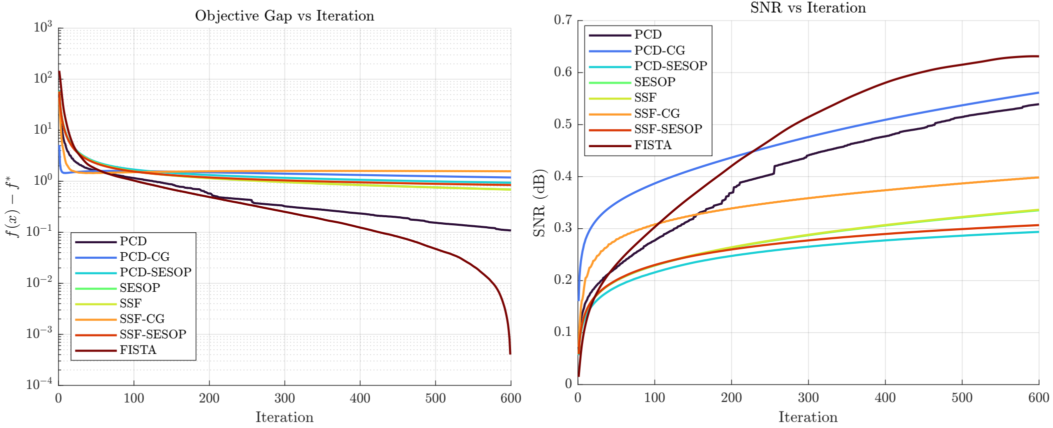

Figure 7: Evaluations of reconstructions for experiment 4 in case of $\lambda = 1 \times 10^{-3}$

Synthetic Experiment with Loris Data: Result(2)

In case of $\lambda = 1 \times 10^{-6}$.

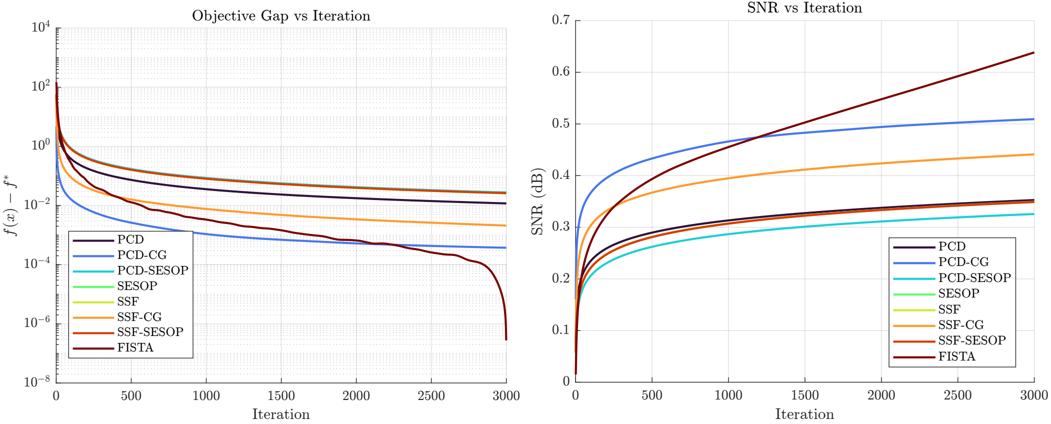

Figure 8: Evaluations of reconstructions for experiment 4 in case of $\lambda = 1 \times 10^{-6}$

Section 4

Conclusion

Conclusion

In conclusion,

- algorithms like PCD, SESOP, SSF, and FISTA are applicable to solve inverse problems.
- each algorithm has its strength and weakness as shown in the convergence/divergence behavior.
- we can measure the efficiency by objective function gap and SNR.

Project Plans

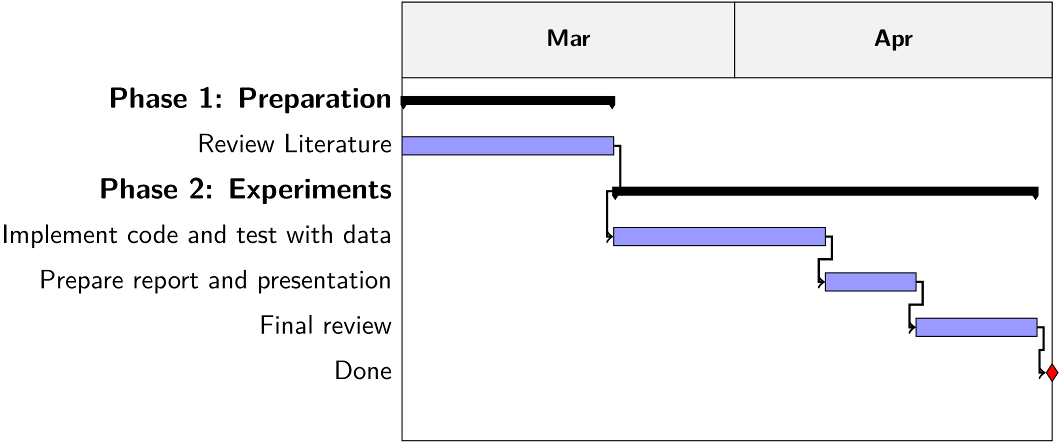

Table 1: Gantt Chart of Work Plan

Problems and Solution

1. The result from experiments and the paper are not the same.
   - vary parameters in the algorithms.
2. Some algorithms consume high computation times.
   - sample data points to be executed.
3. Some algorithms display unstable behaviors.
   - using more robust algorithms.
   - employing techniques like damping or step-size control.

Further Research

1. Measure the performance of other optimization algorithms.
2. Try with some deep learning-based methods.
3. Incorporate other quantitative metrics.
4. Evaluate the computational complexity and execution time of the algorithms.
5. Explore the application of heuristic optimization methods to solve inverse problems.