High Dimensional Statistics Project — Sparse projections onto simplex

Joseph-Marie Ngueponwouo, Emmanuel Gardin

May 16, 2025

Contents

1 Introduction **1**

2 Sparse Projections onto the Simplex **2**
- 2.1 Background and Motivation . . . 2
- 2.2 Projection Problems . . . 2
- 2.3 Interests of Simplex and Hyperplane projections . . . 3
- 2.4 Definitions . . . 3
- 2.5 Underlying Problems . . . 3
- 2.6 Projection Algorithms . . . 4

3 Application to Portfolio Optimization **5**
- 3.1 Problem Statement: Sparse Markowitz Portfolio Adjustment . . . 5
- 3.2 Mathematical Formulation . . . 6
- 3.3 Connection to Linear Regression and Experimental Simplification . . . 6
- 3.4 Experimental Setup Details . . . 7
- 3.5 Restricted Isometry Property (RIP) Considerations . . . 7
- 3.6 Simulation Results and Discussion . . . 8
  - 3.6.1 Empirical RIP Constraint Verification . . . 8
  - 3.6.2 M-Variation Experiment . . . 8
  - 3.6.3 Sparsity Experiment . . . 11
  - 3.6.4 Scalability Experiment . . . 12
  - 3.6.5 Noise Robustness Experiment . . . 13
  - 3.6.6 Convergence Experiment and Impact of Initialization . . . 13

4 Conclusion **14**

5 References **15**

1 Introduction

This mini-course covers sparse projections onto the simplex, focusing on applications like portfolio selection with non-convex constraints, based on Kyrillidis et al. (2012) [2]. We use simulations for illustration. High-dimensional data (e.g., genomics, finance) often requires structural assumptions like sparsity (Lecture 3) or constraints like the simplex (e.g., budget allocation) for effective analysis when standard methods fail. Handling the combination of non-convex sparsity requirements with geometric constraints like the simplex presents unique challenges, particularly when standard convex relaxation techniques prove inadequate. This work explores specialized algorithms designed to efficiently compute these projections, providing tools for tackling such constrained high-dimensional problems.

2 Sparse Projections onto the Simplex

2.1 Background and Motivation

Sparse projections onto the simplex are mathematical tools used to solve optimization problems with sparsity and budget constraints. These projections are particularly useful in fields such as quantum tomography, sparse density estimation, and portfolio optimization. In high-dimensional settings where the number of features p can exceed the number of samples n ($p > n$), problems like linear regression become ill-posed or underdetermined (Lecture 3). Without additional structure, identifying meaningful solutions is impossible. Sparsity ($\|\boldsymbol{\beta}\|_0 \leq k$) provides such structure, assuming only a small subset of features are relevant.

2.2 Projection Problems

The paper addresses two main sparse projection problems:

1. **Simplex Projection**: Finding a Euclidean projection of a vector onto the intersection of sparse vectors and the simplex.
   Given $\mathbf{w} \in \mathbb{R}^p$, find a Euclidean projection of $\mathbf{w}$ onto the intersection of the set of k-sparse vectors $\Sigma_k = \{\boldsymbol{\beta} \in \mathbb{R}^p : |\{i : \beta_i \neq 0\}| \leq k\}$ and the simplex $\Delta_\lambda^+ = \{\boldsymbol{\beta} \in \mathbb{R}^p : \beta_i \geq 0, \sum_i \beta_i = \lambda\}$:

$$\mathcal{P}(\mathbf{w}) \in \underset{\boldsymbol{\beta} \in \Sigma_k \cap \Delta_\lambda^+}{\operatorname{argmin}} \|\boldsymbol{\beta} - \mathbf{w}\|_2 \qquad (1)$$

2. **Hyperplane Projection**: Replacing the simplex constraint with a hyperplane constraint.
   Replace Δ_λ^+ in (1) with the hyperplane constraint $\Delta_\lambda = \{\boldsymbol{\beta} \in \mathbb{R}^p : \sum_i \beta_i = \lambda\}$.

$$\mathcal{P}(\mathbf{w}) \in \underset{\boldsymbol{\beta} \in \Sigma_k \cap \Delta_\lambda}{\operatorname{argmin}} \|\boldsymbol{\beta} - \mathbf{w}\|_2 \qquad (2)$$

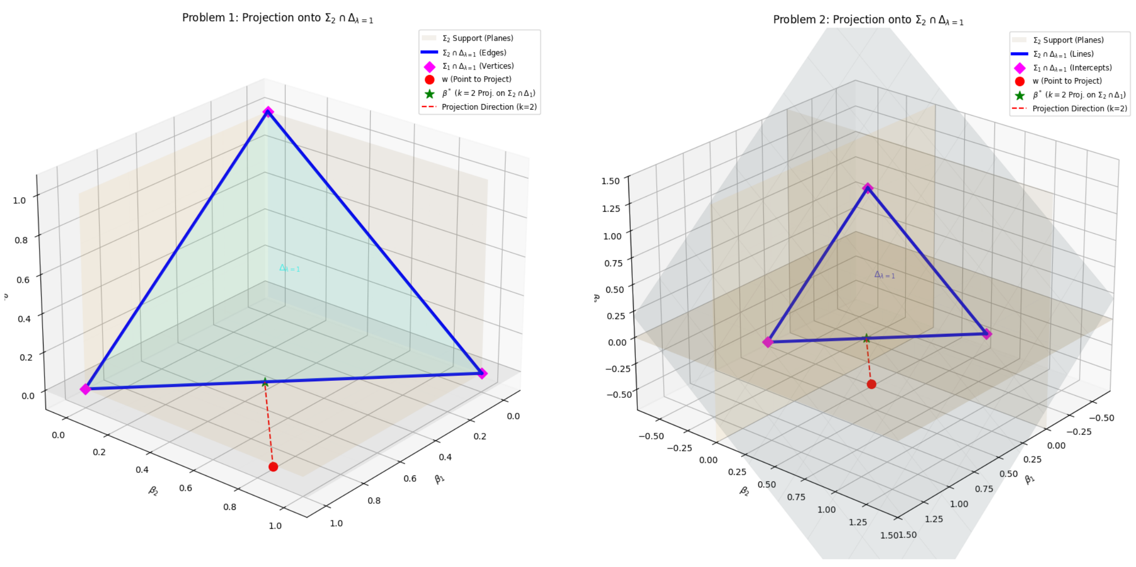

Figure 1: Visualizations of the feasible sets: (left) Projection onto $\Sigma_2 \cap \Delta_\lambda^+$; (right) Projection onto $\Sigma_2 \cap \Delta_\lambda$.

2.3 Interests of Simplex and Hyperplane projections

These projections become interesting when convex relaxations conflict with problem constraints.

$$\min_{\beta\in\mathbb{R}^n:\|\beta\|_0\le s} f(\beta) \xrightarrow{\text{Convexify...}} \min_{\beta\in\mathbb{R}^n:\|\beta\|_1\le\tau} f(\beta)$$

(i) $f(\beta) = ||\boldsymbol{\beta} - \mathbf{w}||_2^2$

(ii) $\|\beta\|_0$: ℓ_0-"norm" where its convex relaxation its $\|\beta\|_1$.

The standard approach to handle the computationally challenging ℓ_0-norm constraint ($\|\boldsymbol{\beta}\|_0 \le s$) is through convex relaxation, typically replacing it with the ℓ_1-norm ($\|\boldsymbol{\beta}\|_1 \le \tau$), as seen in methods like Basis Pursuit (BP) and LASSO (Lecture 4). This relaxation makes the problem convex and often computationally tractable, with theoretical guarantees often relying on conditions like the Restricted Isometry Property (RIP) (Lecture 4) or the Restricted Eigenvalue (RE) condition (Lecture 5).
However, when the problem already involves constraints related to the ℓ_1-norm, such as the simplex constraint $\Delta_\lambda^+ = \{\boldsymbol{\beta}\in\mathbb{R}^p : \beta_i \ge 0, \sum_i \beta_i = \lambda\}$, which implies $\|\boldsymbol{\beta}\|_1 = \lambda$ for $\lambda > 0$, the standard ℓ_1 relaxation loses its effectiveness. Minimizing $\|\boldsymbol{\beta}\|_1$ subject to $\|\boldsymbol{\beta}\|_1 = \lambda$ is trivial.

$$\min_{\beta\in\mathbb{R}^n:\|\beta\|_0\le s, \|\beta\|_1=\lambda} f(\beta) \xrightarrow{\text{Convexify...}} ???$$

This necessitates methods that directly handle the ℓ_0 constraint alongside the simplex or hyperplane constraints, motivating the study of the projection problems (1) and (2). The solutions of (1) and (2) could help us in this type of situation where we cannot benefit from convex relaxation as the problem features convex norms constraints in addition to the non-convex sparsity constraints.
E.g.: Simplex Constraint Δ_1^+

$$\sum_i \beta_i = 1 \quad \text{and} \quad \beta_i \ge 0 \quad \Rightarrow \quad \|\beta\|_1 = 1.$$

$$\min_{\beta\in\mathbb{R}^n:\|\beta\|_0\le s, \|\beta\|_1=1} f(\beta) \xrightarrow{\text{Convexify...}} ???$$

2.4 Definitions

Definition 2.1 (Operator $\mathcal{P}_{L_k}$) We define $\mathcal{P}_{L_k}(\mathbf{w})$ as the operator that keeps the k-largest entries of $\mathbf{w}$ (not in magnitude) and sets the rest to zero. This operation can be computed in $O(p\min(k, \log(p)))$-time (by using MinHeap algorithm for example).
Definition 2.2 (Euclidean projection $\mathcal{P}_{\lambda+}$) The projector onto the simplex is given by

$$(\mathcal{P}_{\lambda^+}(w))_i = [w_i - \tau]_+, \text{ where } \tau := \frac{1}{\rho}\left(\sum_{i=1}^{\rho} w_i - \lambda\right)$$

for $\rho := \max\left\{j : w_j > \frac{1}{j}\left(\sum_{i=1}^{j} w_i - \lambda\right)\right\}$.
Definition 2.3 (Euclidean projection $\mathcal{P}_\lambda$) The projector onto the extended simplex is given by

$$(\mathcal{P}_\lambda(w))_i = w_i - \tau, \text{ where } \tau = \frac{1}{p}\left(\sum_{i=1}^{p} w_i - \lambda\right).$$

2.5 Underlying Problems

Let β^* be a projection of $\mathbf{w}$ onto $\Sigma_k \cap \Delta_\lambda^+$ or $\Sigma_k \cap \Delta_\lambda$.
Remark 1 The Problem 1 and 2 statements can be equivalently transformed into the following nested minimization problem:

$$\{S^*, \beta^*\} = \mathrm{argmin}_{S:S\in\Sigma_k}\left[\mathrm{argmin}_{\substack{\beta:\beta_S\in\Delta_\lambda^+ \text{ or } \Delta_\lambda,\\ \beta_{S^c}=0}} \|(\beta-\mathbf{w})_S\|_2^2 + \|\mathbf{w}_{S^c}\|_2^2\right],$$

where $\mathrm{supp}(\beta^*) = S^*$ and $\beta^* \in \Delta_\lambda^+$ or Δ_λ.
Therefore, given $S^* = \mathrm{supp}(\beta^*)$, we can find β^* by projecting $\mathbf{w}_{S^*}$ onto Δ_λ^+ or Δ_λ within the k-dimensional space. Thus, the difficulty is finding S^*. Hence, **we split the problem into the task of finding the support and then finding the values on the support**. This decomposition into finding the optimal support S^* and then determining the values $\boldsymbol{\beta}^*|_{S^*}$ via projection ($\mathcal{P}_{\lambda^+}$ or $\mathcal{P}_\lambda$) mirrors the conceptual approach of Best Subset Selection (Lecture 3), although finding S^* directly is generally NP-hard. The GSSP and GSHP algorithms provide efficient heuristics or exact solutions for these specific projection problems.
Given any support S, the unique corresponding estimator is $\widehat{\beta}|_S = \mathcal{P}_{\lambda^+}(\mathbf{w}|_S)$. We conclude that β^* satisfies $\beta^*_{(S^*)^c} = 0$ and $\beta^*_{|S^*} = \mathcal{P}_{\lambda^+}(\mathbf{w}|_{S^*})$, and

$$S^* \in \mathrm{argmin}_{S:S\in\Sigma_k} \|\mathcal{P}_{\lambda^+}(\mathbf{w}|_S) - \mathbf{w}|_S\|_2^2 + \|\mathbf{w}|_{S^c}\|_2^2$$

$$= \mathrm{argmax}_{S:S\in\Sigma_k} F_+(S)$$

where $F_+(S) := \sum_{i\in S}\left(w_i^2 - ((\mathcal{P}_{\lambda^+}(\mathbf{w}|_S))_i - w_i)^2\right)$.
This set function can be simplified to

$$F_+(S) = \sum_{i\in S}(w_i^2 - \tau^2),$$

where τ (which depends on S) is as in Lemma 1.
Lemma 1 Let $\beta = \mathcal{P}_{\lambda^+}(\mathbf{w})$ where $\beta_i = [w_i - \tau]_+$. Then, $w_i \geq \tau$ for all $i \in S = \mathrm{supp}(\beta)$. Furthermore,

$$\tau = \frac{1}{|S|}\left(\sum_{i\in S} w_i - \lambda\right).$$

Proof Directly from the definition of τ in Definition 2.2. The intuition is quite simple : the "threshold" τ should be smaller than the smallest entry in the selected support, or we unnecessarily shrink the coefficients that are larger without introducing any new support to the solution. Same arguments apply to inflating the coefficients to meet the simplex budget.
Remark 2 Similar to above, we conclude that β^* satisfies $\beta^*_{|S^*} = \mathcal{P}_\lambda(\mathbf{w}_{|S^*})$ and $\beta^*_{|(S^*)^c} = 0$, where

$$S^* \in \underset{S:S\in\Sigma_k}{\mathrm{argmin}}\|\mathbf{z}-\mathbf{w}\|_2 = \underset{S:S\in\Sigma_k}{\mathrm{argmax}}\; F(S) \tag{9}$$

where $\mathbf{z} \in \mathbb{R}^p$ with $\mathbf{z}_{|S} = \mathcal{P}_\lambda(\mathbf{w}_{|S})$ and $\mathbf{z}_{|S^c} = 0$ and

$$F(S) := \left(\sum_{i\in S} w_i^2\right) - \frac{1}{|S|}\left(\sum_{i\in S} w_i - \lambda\right)^2.$$

2.6 Projection Algorithms

Two algorithms are proposed to solve these problems:

- **GSSP (Greedy Selector and Simplex Projector)**: Used for simplex projection.
- **GSHP (Greedy Selector and Hyperplane Projector)**: Used for hyperplane projection.

```
1: $S^* = \text{supp}(\mathcal{P}_{L_k}(\mathbf{w}))$ {Select support}
2: $\beta_{|S^*} = \mathcal{P}_{\lambda^+}(\mathbf{w}|_{S^*}), \beta_{(S^*)^c} = 0$ {Final projection}
```

Algorithm 1 GSSP (Greedy Selector and Simplex Projector) $O(plogp)$

Theorem 1 Algorithm 1 exactly solves Problem 1.

Proof. Intuitively, the k-largest coordinates should be in the solution. To see this, suppose that $\mathbf{u}$ is the projection of $\mathbf{w}$. Let w_i be one of the k-(most positive) coordinates of $\mathbf{w}$ and $u_i = 0$. Also, let $w_j < w_i$, $i \neq j$ such that $u_j > 0$. We can then construct a new vector $\mathbf{u}'$ where $u'_j = u_i = 0$ and $u'_i = u_j$. Therefore, $\mathbf{u}'$ satisfies the constraints, and it is closer to $\mathbf{w}$, i.e.,

$$\|\mathbf{w} - \mathbf{u}\|_2^2 - \|\mathbf{w} - \mathbf{u}'\|_2^2 = 2u_j(w_i - w_j) > 0.$$

Hence, $\mathbf{u}$ cannot be the projection.
To be complete in the proof, we also need to show that the cardinality k solutions are as good as any other solution with cardinality less than k. Suppose there exists a solution $\mathbf{u}$ with support $|S| < k$. Now add *any* elements to S to form $\hat{S}$ with size k. Then consider $\mathbf{w}$ restricted to $\hat{S}$, and let $\hat{\mathbf{u}}$ be its projection onto the simplex. Because this is a projection,

$$\|\hat{\mathbf{u}}_{\hat{S}} - \mathbf{w}_{\hat{S}}\| \leq \|\mathbf{u}_{\hat{S}} - \mathbf{w}_{\hat{S}}\|,$$

hence

$$\|\hat{\mathbf{u}} - \mathbf{w}\| \leq \|\mathbf{u} - \mathbf{w}\|.$$

□

Algorithm 2 GSHP(Greedy Selector and Hyperplane Projector) $O(plog_2(p))$

```
1: $\ell = 1$, $S = j$, $j \in \arg\max_i[\lambda w_i]$ {Initialize}
2: Repeat: $\ell \leftarrow \ell + 1$, $S \leftarrow S \cup j$, where $j \in \arg\max_{i \in N \setminus S} \left| w_i - \frac{\sum_{j \in S} w_j - \lambda}{\ell - 1} \right|$ {Grow}
3: Until $\ell = k$, set $S^* \leftarrow S$ {Terminate}
4: $\beta_{|S^*} = \mathcal{P}_{\lambda}(\mathbf{w}|_{S^*}), \beta_{(S^*)^c} = 0$ {Final projection}
```

Note : GSHP selects the index of the largest element with the same sign as λ (Step 1). It then grows the index set one at a time by finding the farthest element from the current mean, as adjusted by λ (Step 2).
Theorem 2 Algorithm 2 exactly solves Problem 2.

Proof. **Hint** : First describe the proof for the case $k \geq 2$ for $\lambda = 0$ and then explain how it generalizes for $\lambda \neq 0$. Notice also that the goal is to maximize $F(S)$ also written as :

$$F(S) := \left(\sum_{i \in S} w_i^2\right) - \frac{1}{|S|}\left(\sum_{i \in S} w_i - \lambda\right)^2 = \lambda(2w_1 - \lambda) + \sum_{j=2}^{|S|} \frac{j-1}{j}\left(w_j - \frac{\sum_{i=1}^{j-1} w_i - \lambda}{j-1}\right)^2$$

It's clear that the first step of the algorithm is important because for $k = 1$, maximizing $F(S)$ is equivalent to maximizing λw_i. For $k \geq 2$, use the right hand side of the above equation. □

3 Application to Portfolio Optimization

3.1 Problem Statement: Sparse Markowitz Portfolio Adjustment

In practical portfolio management, investors often need to adjust their existing asset allocation $\overline{\boldsymbol{\beta}} \in \mathbb{R}^p$ rather than constructing a portfolio from scratch. This adjustment, $\boldsymbol{\delta}_{\boldsymbol{\beta}}$, results in a new

portfolio $\boldsymbol{\beta} = \overline{\boldsymbol{\beta}} + \boldsymbol{\delta}_{\boldsymbol{\beta}}$. Due to transaction costs (brokerage fees, bid-ask spreads, market impact), it's desirable to make this adjustment using only a small number of assets. This leads to a sparse portfolio adjustment problem within the Markowitz mean-variance framework [1, 3].
The objective is to find an adjustment $\boldsymbol{\delta}_{\boldsymbol{\beta}}$ that minimizes the portfolio variance while achieving a target expected return, subject to constraints on the sparsity of the adjustment and the total budget change.

3.2 Mathematical Formulation

The problem is:

$$\boldsymbol{\delta}^*_{\boldsymbol{\beta}} \in \underset{\boldsymbol{\delta}_{\boldsymbol{\beta}} \in \Sigma_k \cap \Delta_\lambda}{\operatorname{argmin}} \left\{ (\overline{\boldsymbol{\beta}} + \boldsymbol{\delta}_{\boldsymbol{\beta}})^T \boldsymbol{\Sigma} (\overline{\boldsymbol{\beta}} + \boldsymbol{\delta}_{\boldsymbol{\beta}}) - \gamma \boldsymbol{\mu}^T (\overline{\boldsymbol{\beta}} + \boldsymbol{\delta}_{\boldsymbol{\beta}}) \right\} \tag{3.1}$$

where:

- $\boldsymbol{\delta}_{\boldsymbol{\beta}} \in \mathbb{R}^p$: Weight changes.
- $\overline{\boldsymbol{\beta}} \in \mathbb{R}^p$: Current weights.
- $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$: Covariance matrix.
- $\boldsymbol{\mu} \in \mathbb{R}^p$: Expected returns.
- $\gamma > 0$: Risk aversion parameter ($\neq$ projection τ).
- $\Sigma_k = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_0 \leq k\}$: k-sparse vectors (limits traded assets $\leq k$).
- $\Delta_\lambda = \{\boldsymbol{\delta} : \sum_i \delta_i = \lambda\}$: Hyperplane constraint (net adjustment $= \lambda$; $\lambda = 0$ self-financing, $\lambda > 0$ adding capital, $\lambda < 0$ withdrawing).

Problem (3.1) is computationally hard as it requires optimizing over the feasible set $\Sigma_k \cap \Delta_\lambda$, which combines a non-convex sparsity constraint (Σ_k) with a budget constraint (Δ_λ), making the overall set non-convex.

3.3 Connection to Linear Regression and Experimental Simplification

Solving (3.1) directly is difficult. To isolate and evaluate the performance of the sparse projection algorithms (specifically GSHP for the $\Sigma_k \cap \Delta_\lambda$ constraint), the paper [2] adopts a common strategy from compressed sensing literature: they simplify the problem to a linear regression framework where the goal is to recover a known sparse vector satisfying the constraints. This simplification to the form (3.2) casts the problem into a constrained version of the standard linear regression setting discussed in Lecture 1 and Lecture 5. While standard OLS/MLE (Lecture 1) is used when $n > p$ and no sparsity is assumed, and LASSO (Lecture 5) is used when sparsity is desired but without the simplex constraint, this setup allows testing the specific projection onto $\Sigma_k \cap \Delta_\lambda$ within a familiar sparse recovery framework.
This simplification follows the approach in [3], where the Markowitz optimization (without the sparsity constraint Σ_k) is reformulated as a constrained least-squares problem. Minimizing the variance ($\boldsymbol{w}^T \boldsymbol{\Sigma} \boldsymbol{w}$) subject to $\boldsymbol{w}^T \boldsymbol{\mu} = \rho$ and $\boldsymbol{w}^T \mathbf{1} = 1$ can be shown to be related to minimizing $\|\rho \mathbf{1}_T - \mathbf{R}\boldsymbol{w}\|_2^2$, where $\mathbf{R}$ is the $T \times p$ matrix of historical returns over T periods.
For the numerical experiments in [2], the problem is further abstracted to: Find $\boldsymbol{\beta}$ that solves

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \quad \text{subject to} \quad \boldsymbol{\beta} \in \Sigma_k \cap \Delta_\lambda \tag{3.2}$$

Here:

- $\boldsymbol{\beta}$ represents the target portfolio (or adjustment, depending on interpretation).
- $\mathbf{X}$ is a synthetic $m \times p$ measurement matrix, analogous to the historical return matrix $\mathbf{R}$ or a general sensing matrix.
- $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^*$ are the synthetic measurements generated from a *known ground truth* portfolio $\boldsymbol{\beta}^*$.
- $\boldsymbol{\beta}^*$ is constructed beforehand to be k-sparse and satisfy the budget constraint, i.e., $\boldsymbol{\beta}^* \in \Sigma_k \cap \Delta_\lambda$.

This setup allows a clear evaluation: how well can an algorithm recover the known $\boldsymbol{\beta}^*$ when using the sparse projection $\mathcal{P}_{\Sigma_k \cap \Delta_\lambda}$ within an iterative solver like Projected Gradient Descent (PGD)? This effectively tests the quality of the projection itself. In the experiments reported, they use $\lambda = 1$ and generate $\boldsymbol{\beta}^*$ on the standard simplex Δ_1^+, which automatically satisfies $\boldsymbol{\beta}^* \in \Delta_1$.

3.4 Experimental Setup Details

The simulations aim to replicate the experiment in Section 8 of [2], focusing on recovering a synthetic sparse portfolio $\boldsymbol{\beta}^*$ using the formulation (3.2).
Parameters:

- $p = 1000$ (dimension)
- $k = 100$ (sparsity)
- Ground Truth $\boldsymbol{\beta}^* \in \mathbb{R}^p$: Support: k uniform random indices; Values: i.i.d. Uniform(0,1); Normalized: $\sum \beta_i^* = \lambda = 1$ ($\boldsymbol{\beta}^* \in \Sigma_k \cap \Delta_1^+$).
- Matrix $\mathbf{X}$: $m \times p$, i.i.d. $\mathcal{N}(0,1)$ entries.
- Measurements $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^*$ (noiseless initially).
- m: varied (e.g., 280-400).
- Budget $\lambda = 1$.
- Algorithms Compared:
  1. **Convex (SPGL1)** [5]: Solves the Basis Pursuit problem (Lecture 4), finding the minimum ℓ_1-norm solution satisfying the data consistency and budget constraints:
  $$\hat{\boldsymbol{\beta}}_{\text{cvx}} = \operatorname{argmin}_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\|_1 \quad \text{s.t.} \quad \begin{bmatrix} \mathbf{X} \\ \mathbf{1}^T \end{bmatrix} \boldsymbol{\beta} = \begin{bmatrix} \mathbf{y} \\ \lambda \end{bmatrix} \tag{3.3}$$
  (Note: The paper uses $\mathbf{1}^T/\sqrt{p}$ and $\lambda/\sqrt{p}$ for better conditioning, which is equivalent up to scaling). The solution is generally not k-sparse. This is a standard convex relaxation approach to finding sparse solutions.
  2. **Non-convex (PGD+GSHP)**: Solves (3.2) using Projected Gradient Descent (PGD), a standard iterative method, but the projection here is onto the non-convex set $\Sigma_k \cap \Delta_\lambda$:
  $$\boldsymbol{\beta}^{(t+1)} = \mathcal{P}_{\Sigma_k \cap \Delta_\lambda}(\boldsymbol{\beta}^{(t)} - \mu \mathbf{X}^T(\mathbf{X}\boldsymbol{\beta}^{(t)} - \mathbf{y})) \tag{3.4}$$
  where $\mathcal{P}_{\Sigma_k \cap \Delta_\lambda}$ is the GSHP Algorithm 2.
     - Initialization $\boldsymbol{\beta}^{(0)}$: The solution from SPGL1, $\hat{\boldsymbol{\beta}}_{\text{cvx}}$.
     - Step size μ: Typically $1/\|\mathbf{X}^T\mathbf{X}\|_2 \approx 1/\|\mathbf{X}\|_2^2$.
     - Stopping criteria: Max iterations (e.g., 1000) or small relative change (e.g., 10^{-6}).
- Performance Metric: Median relative ℓ_2 recovery error $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*\|_2 / \|\boldsymbol{\beta}^*\|_2$ over multiple Monte Carlo trials (e.g., $N = 30$).

3.5 Restricted Isometry Property (RIP) Considerations

The Restricted Isometry Property (RIP), introduced in Lecture 4 as a sufficient condition for guarantees in ℓ_1-minimization (Basis Pursuit), plays a similar role in guaranteeing performance for various iterative algorithms for sparse recovery, including variants of PGD [6]. It essentially ensures that the matrix $\mathbf{X}$ acts almost like an isometry when restricted to sparse vectors.
Definition 2.4 (RIP): A matrix $\mathbf{X}$ satisfies the k-RIP with constant $\delta_k \in [0, 1)$ if

$$(1 - \delta_k)\|\boldsymbol{\beta}\|_2^2 \leq \|\mathbf{X}\boldsymbol{\beta}\|_2^2 \leq (1 + \delta_k)\|\boldsymbol{\beta}\|_2^2$$

holds for all k-sparse vectors $\boldsymbol{\beta}$. (Often $\mathbf{X}$ is normalized such that $E[\|X\beta\|_2^2] \approx \|\beta\|_2^2$).
Relevance to PGD+GSHP: The analysis in [2] (Section 2, guarantees of gradient scheme (3)) notes that if $\mathbf{X}$ satisfies RIP for $2k$-sparse or $3k$-sparse vectors with a sufficiently small constant (e.g., $\delta_{2k} < 1/3$ or $\delta_{3k} < 1/2$ for related algorithms), then PGD-like methods can converge to the true solution $\boldsymbol{\beta}^*$.
RIP in the Experimental Setup:

- **Theoretical Guarantee**: Random Gaussian matrices $\mathbf{X}$ (like the one used) are known to satisfy RIP with high probability if the number of measurements m is sufficiently large, specifically $m = \Omega(k \log(p/k)/\delta_k^2)$.
- **Practical Verification**: Computing the exact RIP constant δ_k is NP-hard. We can, however, estimate it empirically by testing many random k-sparse vectors (see Section 3.6.1).

3.6 Simulation Results and Discussion

We now present the results from various experiments designed to test the PGD+GSHP algorithm's performance under different conditions.

3.6.1 Empirical RIP Constraint Verification

As discussed in Section 3.5, the convergence theory for PGD-like algorithms relies on RIP constants δ_{2k} or δ_{3k} being sufficiently small. We empirically estimated these constants for our main experimental setting ($p = 1000, k = 100, m = 280$) using 10000 random sparse vectors. The results are summarized in Table 1.

Table 1: Empirical RIP Constant Estimates ($p = 1000, k = 100, m = 280$)

Sparsity Level (s)	Order	Estimated δ_s	Theoretical Threshold	Condition Met?
100	k	0.3764	-	-
200	$2k$	0.3819	$< 1/3$	**No**
300	$3k$	0.4334	$< 1/2$	Yes

Findings: The results show that $\delta_{2k} \approx 0.38$, which does not satisfy the stricter $\delta_{2k} < 1/3$ condition required for guaranteed monotonic descent or fast convergence in PGD (condition related to Eq. 5 in [2]). However, $\delta_{3k} \approx 0.43$ does satisfy the $\delta_{3k} < 1/2$ condition, which can be sufficient for proving convergence to a neighborhood of the true solution (related to Eq. 6 in [2]).

Implication: The fact that the strong RIP condition ($\delta_{2k} < 1/3$) is not empirically met, especially at this relatively low measurement rate $m/p = 0.28$ (which is near the theoretical $m \approx k \log(p/k)$ limit), helps explain why the non-convex PGD+GSHP, while generally outperforming SPGL1, exhibits convergence issues for lower values of m.

Real Data Caveat: In real portfolio optimization, the matrix $\mathbf{X}$ (related to asset returns) is unlikely to be random Gaussian. Asset returns often exhibit heavy tails and dependencies, meaning RIP is even less likely to hold theoretically.

3.6.2 M-Variation Experiment

Goal: Evaluate solver performance (recovery error) as the number of measurements m varies ($m \in [280, 400]$), keeping $p = 1000$ and $k = 100$ constant.

Findings:

- The non-convex PGD+GSHP approach consistently achieves a lower median relative error compared to the convex SPGL1 solution across the range of m.

- The error for both methods decreases as m increases, as expected.
- The variance (observed from boxplots for a single m, e.g., $m = 280$) in the error of the non-convex solver is relatively higher for lower values of m, but decreases as m increases. The SPGL1 error variance appears more stable.
- Coefficient recovery plots for a single trial show that PGD+GSHP recovers coefficients closer to the ground truth $\boldsymbol{\beta}^*$ and respects the sparsity $k = 100$. SPGL1 produces a dense solution with many small non-zero coefficients deviating significantly from $\boldsymbol{\beta}^*$.
- Support recovery plots confirm that PGD+GSHP correctly identifies most of the true support, while SPGL1 identifies almost all indices as part of the support due to its dense nature.

Interpretation: The superiority of PGD+GSHP highlights the benefit of enforcing the exact sparsity constraint. The decreasing error as m increases aligns with the theory that RIP conditions (Lecture 4) become better satisfied with more measurements, leading to more stable recovery. The higher variance at low m for PGD+GSHP is likely due to the RIP condition being less well-satisfied, potentially leading to convergence to local minima or slower convergence in some trials. As m increases, RIP improves, leading to more robust performance. SPGL1, being convex, is less sensitive in this regard but provides a less accurate (dense) solution.

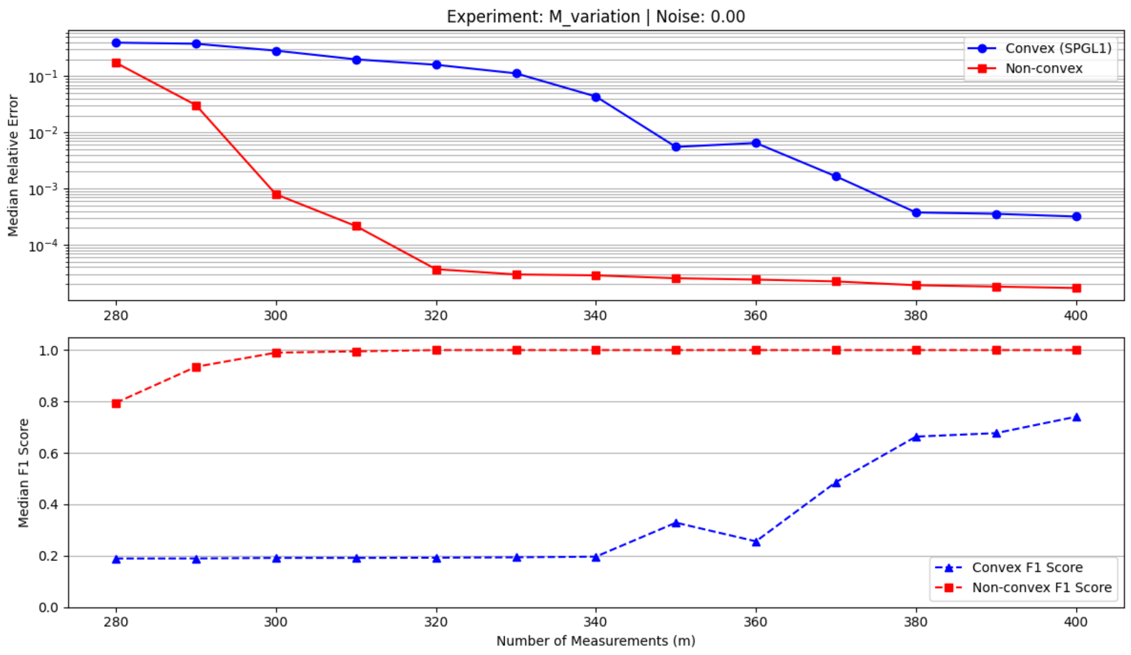

Figure 2: M-Variation: Median Relative Error (top) and Median F1-Score (bottom) vs. Number of Measurements (m) ($p = 1000, k = 100$). Compares PGD+GSHP and SPGL1.

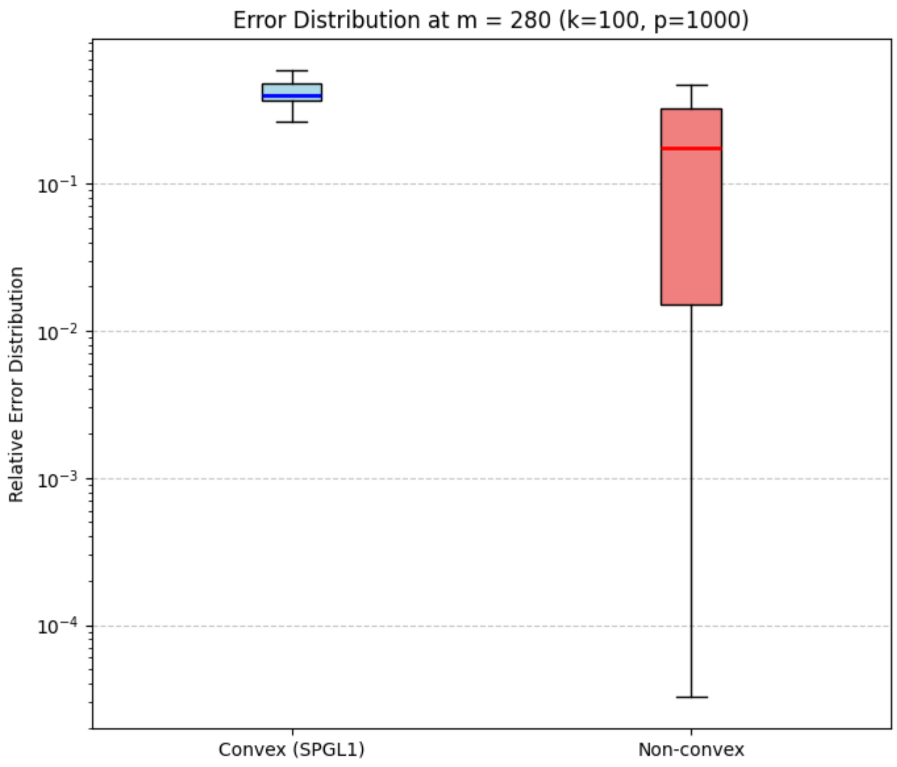

Figure 3: M-Variation: Distribution of Relative Errors for $m = 280$ $(p = 1000, k = 100)$ across multiple trials. Compares PGD+GSHP and SPGL1.

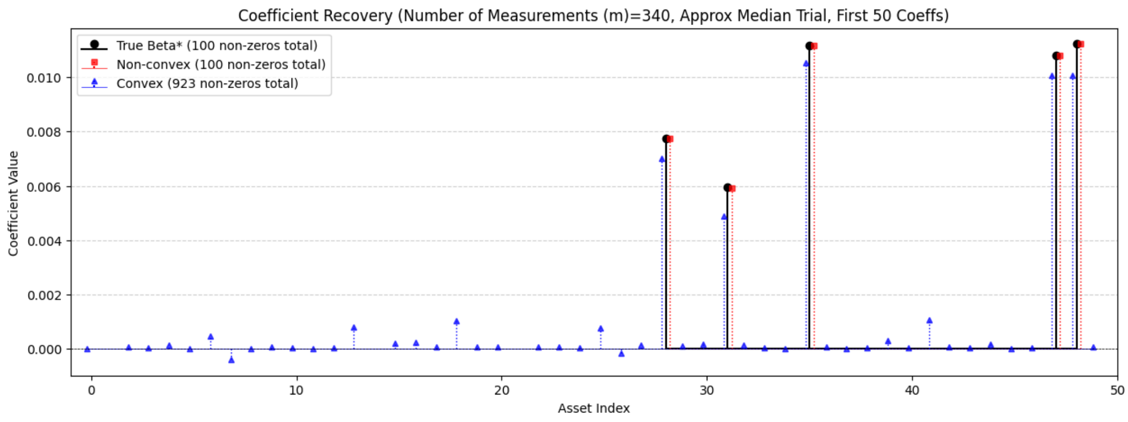

Figure 4: M-Variation: Example Coefficient Recovery for a single trial $(m = 340, p = 1000, k = 100)$. First 50 coefficients. Shows Ground Truth (β^*), PGD+GSHP result, and SPGL1 result.

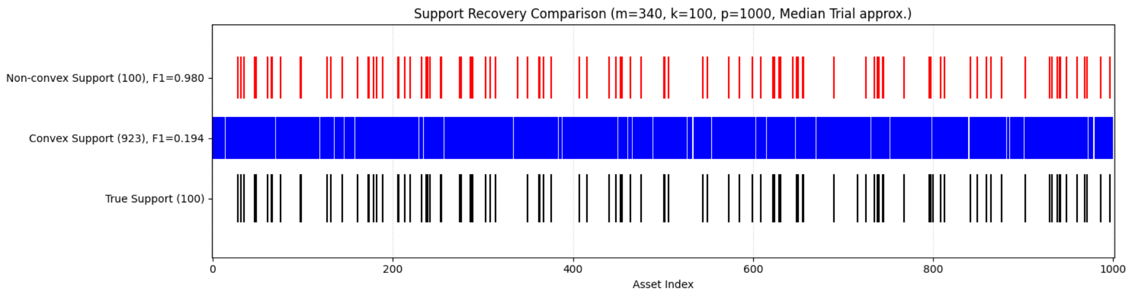

Figure 5: M-Variation: Example Support Recovery for a single trial ($m = 340$, $p = 1000, k = 100$). Shows the indices of non-zero coefficients for Ground Truth (β^*), PGD+GSHP result, and SPGL1 result.

3.6.3 Sparsity Experiment

Goal: Evaluate solver performance as the sparsity level k varies (e.g., $k \in [|50, 200|]$), keeping $p = 1000$ and m (e.g., $m = 290$) constant.

Findings:

- As the sparsity level k increases (signal becomes denser), the median relative error increases for both solvers.
- The performance gap between PGD+GSHP and SPGL1 narrows as k increases.
- For sufficiently large k (relative to m and p), the median error of the non-convex solver might even exceed that of the convex solver.

Interpretation: Recovery becomes harder as the signal gets denser for a fixed number of measurements m. The theoretical requirement $m = \Omega(k \log(p/k))$ indicates that as k grows, more measurements are needed for reliable recovery. When this condition is poorly met (large k), RIP conditions (Lecture 4) degrade significantly, negatively impacting the non-convex solver which relies more heavily on it. SPGL1's performance also degrades but might be more robust when the problem is far from the ideal sparse recovery regime.

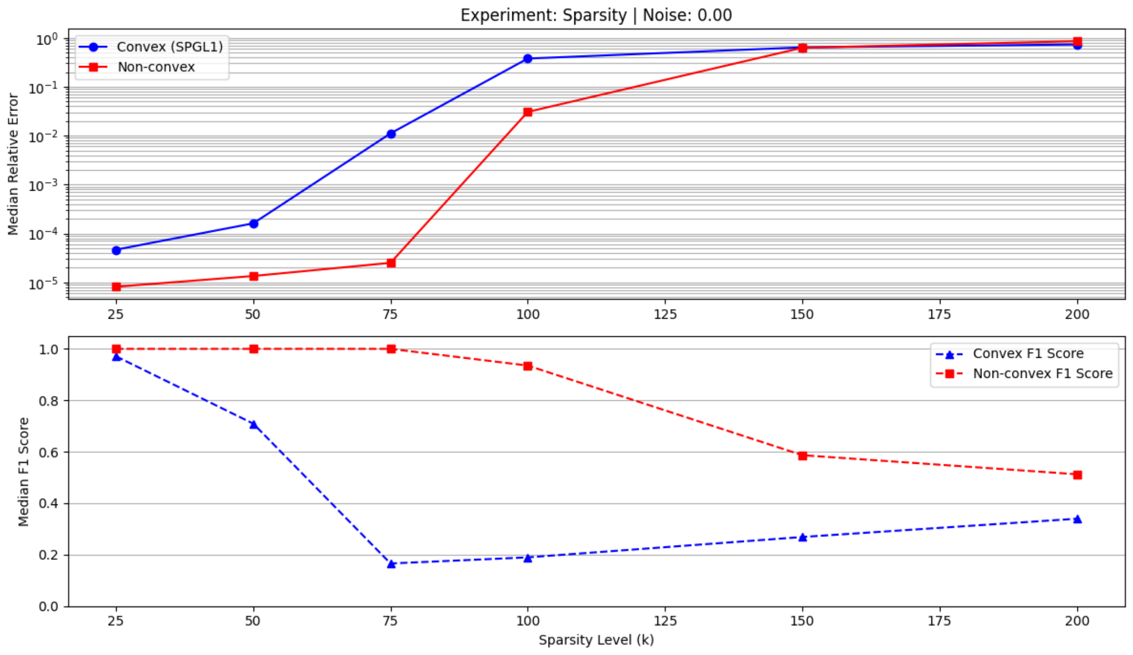

Figure 6: Median Relative Error vs. Sparsity Level k ($p = 1000, m = 290$).

3.6.4 Scalability Experiment

Goal: Evaluate solver performance and runtime as the problem dimension p increases (e.g., $p \in [500, 2500]$), while keeping the ratios k/p (e.g., $k/p = 0.1$) and m/p (e.g., $m/p = 0.3$) constant to maintain consistent problem difficulty.

Findings:

- The median relative error for the convex SPGL1 solver tends to remain constant as p increases under constant ratios, while the error for the PGD+GSHP solver decreases.
- The median runtime for the non-convex PGD+GSHP solver increases with p. Separate timing of the GSHP projection step within the solver shows that it accounts for the vast majority of the total runtime (Figure 7, bottom).
- A curve of the form $C \cdot p \log p$, fitted to the median *projection time*, demonstrates a good fit, empirically validating the expected quasi-linear time complexity of the GSHP algorithm.

Interpretation: The evolution of the error for the non-convex solver can be explained by the fact that the theoretical requirement $m = \Omega(k \log(p/k))$ becomes better satisfied as p increases and $k/p = 0.1$ remains constant. The runtime analysis (Figure 7, bottom) provides further insights into the non-convex solver's scalability. The total runtime is clearly dominated by the time spent within the GSHP projection step in each PGD iteration. The time for gradient computation (primarily matrix-vector multiplications) appears negligible in comparison, likely due to highly optimized numerical linear algebra routines (e.g., BLAS used by NumPy). The satisfactory fit of the $C \cdot p \log p$ curve to the empirical projection times supports the theoretical quasi-linear time complexity ($O(p \log p)$ of the GSHP algorithm when sorting dominates, confirming its computational feasibility and predictable scaling for large problem dimensions.

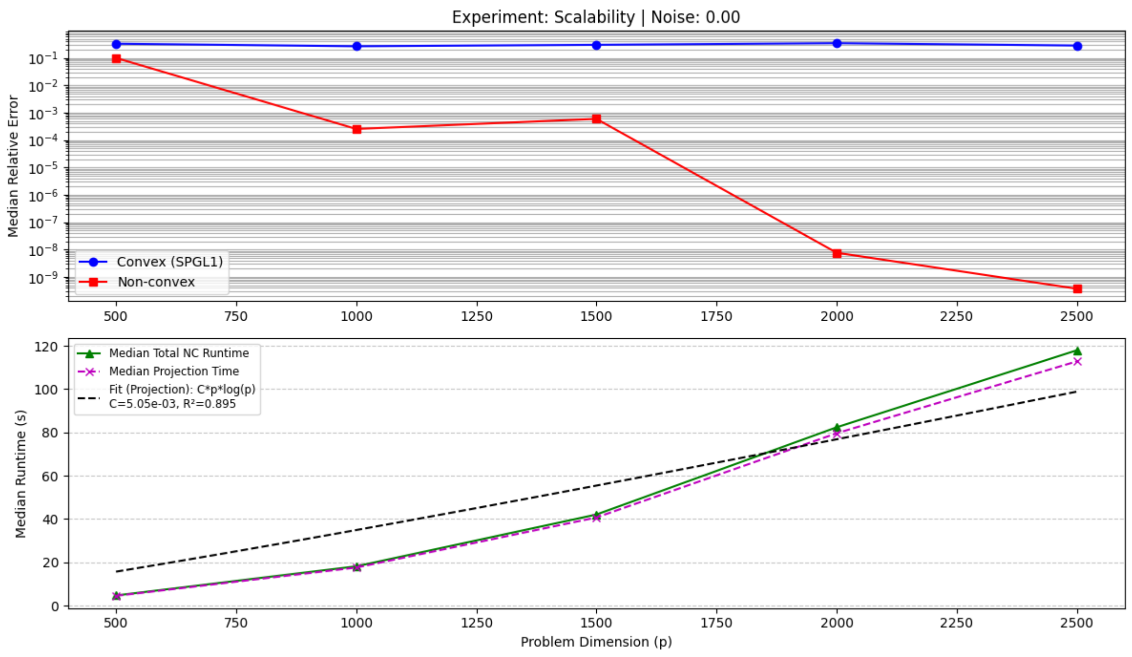

Figure 7: Scalability experiment results ($k/p = 0.1$, $m/p = 0.3$). Top: Median relative recovery error for PGD+GSHP and SPGL1 vs. problem dimension p. Bottom: Median runtime analysis vs. p, showing total PGD+GSHP time, time for the GSHP projection step only, and a fitted curve $C \cdot p \log p$ to the projection time.

3.6.5 Noise Robustness Experiment

Goal: Evaluate performance while varying levels of additive Gaussian noise $\boldsymbol{\eta}$ on the measurements, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\eta}$, where $\eta_i \sim \mathcal{N}(0, \sigma^2)$. Keep p, k, m constant.

Findings:

- As the noise level (σ) increases, the recovery error increases for both solvers, as expected.
- The non-convex PGD+GSHP solver still generally outperforms the SPGL1 (run in BPDN mode for noise) baseline, but the gap may narrow at higher noise levels.

Interpretation: Both algorithms exhibit progressive degradation with increasing noise. The PGD+GSHP method, which uses projections based directly on the (noisy) gradient, can still leverage the sparsity structure effectively. SPGL1, when used in Basis Pursuit Denoise (BPDN) mode (minimizing $\|\boldsymbol{\beta}\|_1$ subject to $\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2 \leq \epsilon$), also handles noise but doesn't enforce the exact sparsity k.

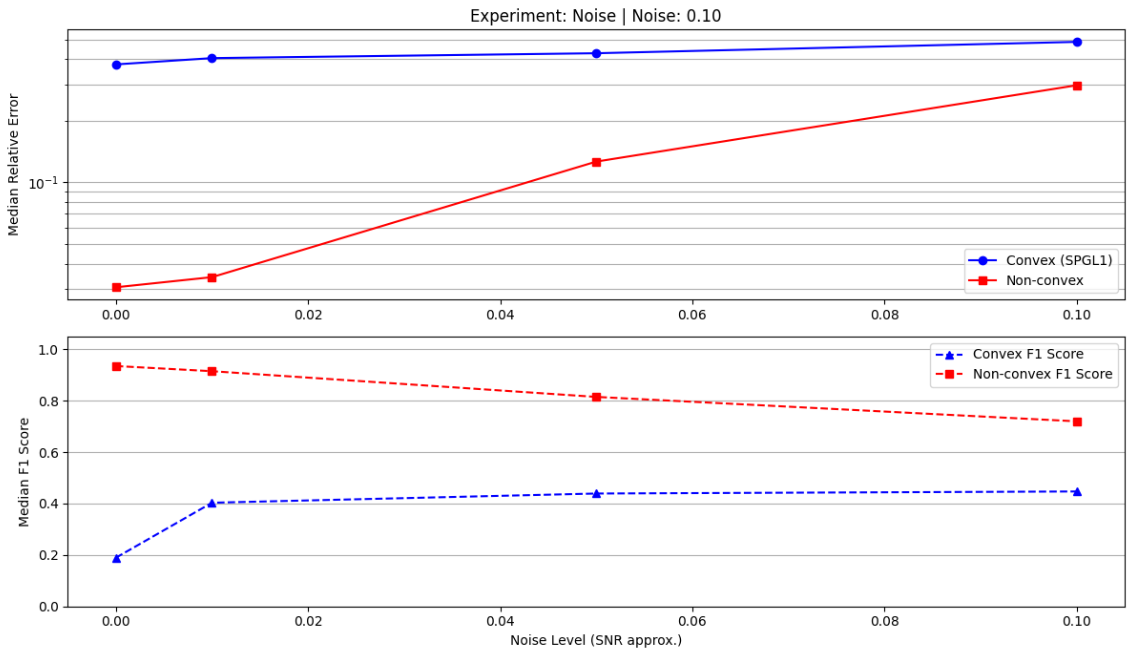

Figure 8: Median Relative Error vs. Noise Standard Deviation σ ($p = 1000, k = 100, m = 290$).

3.6.6 Convergence Experiment and Impact of Initialization

Goal: Analyze the convergence of the PGD+GSHP solver for a fixed problem instance, particularly focusing on how different initialization strategies $\boldsymbol{\beta}^{(0)}$ affect the convergence path and the final solution quality. Keep p, k, m constant. Track objective function value $f(\boldsymbol{\beta}^{(t)}) = \frac{1}{2}\|\mathbf{X}\boldsymbol{\beta}^{(t)} - \mathbf{y}\|_2^2$ and relative solution change $\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}\|_2 / \|\boldsymbol{\beta}^{(t)}\|_2$ per iteration. Initialization strategies compared include:

- **SPGL1**: Initialization with the dense solution $\hat{\boldsymbol{\beta}}_{\text{cvx}}$ obtained from the convex SPGL1 solver.
- **Zero**: Initialization with the zero vector, $\boldsymbol{\beta}^{(0)} = \mathbf{0}$.
- **Random Projection**: Initialization by projecting a random Gaussian vector onto the budget hyperplane Δ_λ (this initial point is dense but satisfies the sum constraint).
- **Gradient Step**: Initialization by taking one gradient descent step from the zero vector, $\boldsymbol{\beta}^{(1)} = \mathcal{P}_{\Sigma_k \cap \Delta_\lambda}(-\mu \nabla f(\mathbf{0})) = \mathcal{P}_{\Sigma_k \cap \Delta_\lambda}(\mu \mathbf{X}^T \mathbf{y})$, and then projecting onto the non-convex feasible set $\Sigma_k \cap \Delta_\lambda$.

Findings:

- The PGD+GSHP solver converges (relative change goes to zero) for all tested initializations.
- However, due to the non-convexity of the problem (3.2), the solver often converges to different final solutions (different objective values and errors) depending on the starting point $\boldsymbol{\beta}^{(0)}$.
- Initializing with the SPGL1 solution $\hat{\boldsymbol{\beta}}_{\text{cvx}}$ consistently leads to convergence to the solution with the lowest final recovery error (closest to $\boldsymbol{\beta}^*$).
- Zero initialization or random initialization often leads to significantly worse results, indicating convergence to poorer local minima.
- The convergence traces (relative change vs. iteration), are not always monotonic. There are jumps, particularly visible in the relative change plot.

Interpretation: The convergence plots visually confirm the non-convexity of the optimization landscape and the critical impact of initialization. Initializing with the SPGL1 solution, provides a good initial estimate that is relatively close to the desired solution region (satisfying data consistency and budget constraints). Starting from this informed "warm start" helps guide the PGD+GSHP towards a better quality sparse solution, mitigating issues arising from the lack of strong RIP guarantees (Lecture 4). This approach contrasts with the notion of implicit regularization via Gradient Descent from zero initialization (Lecture 6), as here an explicit non-convex projection combined with an informed initialization is necessary due to the complex feasible set $\Sigma_k \cap \Delta_\lambda$. The observed non-monotonic convergence behavior, particularly visible as spikes in the relative change plot (Fig 9, bottom), stems from the discrete nature of the non-convex GSHP projection $\mathcal{P}_{\Sigma_k \cap \Delta_\lambda}$. To enforce the k-sparsity and budget constraints, the projector might abruptly change the support set, for instance, setting one coefficient to 0 while activating another. Such discrete changes can lead to a large relative step $\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}\|_2 / \|\boldsymbol{\beta}^{(t)}\|_2$ even after a small gradient update, causing the observed spikes, especially when RIP conditions (Lecture 4) are weak.

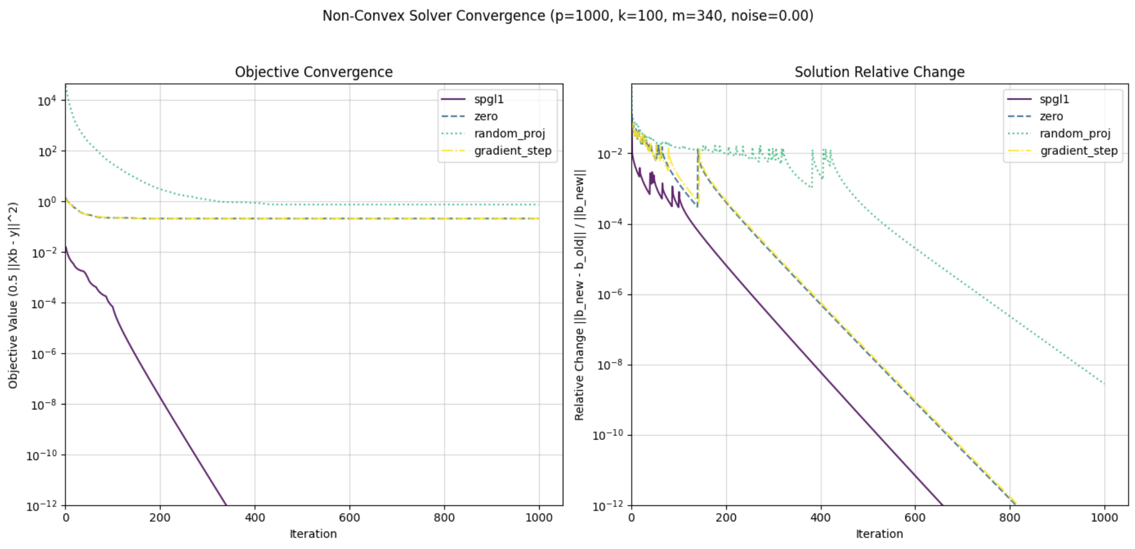

Figure 9: Convergence traces for PGD+GSHP starting from different initializations (SPGL1, Zero, Random Projection, Gradient Step) for $p = 1000, k = 100, m = 340$, Noise=0.00. Top: Objective function value $(0.5\|\mathbf{X}\boldsymbol{\beta}^{(t)} - \mathbf{y}\|_2^2)$ vs. iteration. Bottom: Relative solution change $(\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}\|_2 / \|\boldsymbol{\beta}^{(t)}\|_2)$ vs. iteration.

4 Conclusion

GSSP/GSHP offer efficient (quasilinear complexity in time $O(p \log p)$) exact sparse projections onto simplex/hyperplane, valuable when standard convex relaxations (Lecture 4) fail due to combined

sparsity and budget constraints. Simulations showed PGD+GSHP can refine convex solutions (e.g., SPGL1), achieving lower error by strictly enforcing constraints. This non-convex approach works empirically, especially with good initialization, even under weak RIP conditions, connecting sparse recovery theory to practical non-convex optimization for high-dimensional problems.
However, GSSP/GSHP don't seem to be widely used in practice for portfolio optimization. Potential limitations include:

- **Non-convex Optimization Issues:** The overall PGD+GSHP scheme is non-convex, sensitive to initialization, lacks global guarantees, and requires pre-specifying the sparsity level k, which is often unrealistic.
- **Limited Transaction Cost Model:** GSHP enforces ℓ_0 sparsity (number of trades), but doesn't directly handle common proportional (ℓ_1) costs, which are often better addressed via easier convex ℓ_1 penalties.
- **Incompatibility with Richer Constraints:** GSHP projection is specific to $\Sigma_k \cap \Delta_\lambda$. Real-world portfolio problems can include additional constraints, making the GSHP projection step difficult or inapplicable. Standard convex solvers or alternative heuristics can handle these richer constraint sets more readily.

Despite limited practical adoption, GSSP/GSHP provide valuable insights into handling specific non-convex constraints combinations efficiently.

5 References

References

[1] Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, *7*(1), 77-91.

[2] Kyrillidis, A., Becker, S., Cevher, V., & Koch, C. (2012). Sparse projections onto the simplex. *arXiv preprint arXiv:1206.1529.*

[3] Brodie, J., Daubechies, I., De Mol, C., Giannone, D., & Loris, I. (2009). Sparse and stable Markowitz portfolios. *Proceedings of the National Academy of Sciences*, *106*(30), 12267-12272.

[4] DeMiguel, V., Garlappi, L., Nogales, F. J., & Uppal, R. (2009). A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, *55*(5), 798-812.

[5] van den Berg, E., & Friedlander, M. P. (2008). Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, *31*(2), 890-912.

[6] Candès, E. J., Romberg, J., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, *52*(2), 489-509.