NumOps Project 3: Lasso Optimization Algorithms

Youniss Kandah
JKU Linz

August 2025

Abstract

This report documents the implementation and analysis of three optimization algorithms for Lasso-type problems: Forward-Backward (FB), Projected Gradient (PG), and Active-Set Method (ASM). The project focuses on approximating the sine function over $[-2\pi, 2\pi]$ using polynomial regression while encouraging sparsity in the coefficient vector. We implement these algorithms and compare their performance on both penalized and constrained Lasso formulations, including analysis of condition numbers and pre-conditioning strategies.

Contents

1 Introduction

1.1 Development Environment

The project is implemented in Python 3.13 using modern development tools:

- **Package Manager**: UV for fast dependency resolution
- **Virtual Environment**: Isolated Python environment for reproducibility
- **Core Dependencies**: NumPy, Matplotlib, SciPy, Pandas
- **Development Tools**: pytest, black, flake8

1.2 Project Structure

The codebase is organized into modular components:

```
num_opt_proj_3/
 src/
    algorithms/     # Optimization algorithms (FB, PG, ASM)
    problems/       # Problem formulations
    utils/          # Utility functions
    experiments/    # Experiment scripts
 results/
    plots/          # Generated plots
    data/           # Numerical results
 docs/               # Documentation and reports
 Makefile            # Build and run commands
 pyproject.toml      # Project configuration
```

2 Core Implementation

2.1 Problem Formulation

The core problem formulation has been implemented in the `SineApproximationProblem` class. This class encapsulates:

- **Vandermonde Matrix Construction**: Creates matrix A where $A[j, i] = a_j^i$ for sample points a_j and powers $i = 0, 1, \ldots, n$
- **Objective Function**: Implements $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ with gradient $\nabla f(x) = A^\top(Ax - b)$ and Hessian $\nabla^2 f(x) = A^\top A$
- **Problem Properties**: Computes Lipschitz constant $L = \|A^\top A\|_2$ and condition number for analysis
- **Visualization**: Methods to plot the sine function approximation and compare with true values

2.2 Implementation Details

The problem setup follows the mathematical formulation exactly:

- Sample points: $a_j \in [-2\pi, 2\pi]$ uniformly spaced
- Target values: $b_j = \sin(a_j)$
- Polynomial basis: $\phi(x;t) = \sum_{i=0}^{n} x_i t^i$
- Matrix dimensions: $A \in \mathbb{R}^{m\times(n+1)}$ where $m = 100$ samples and n is the polynomial degree

2.3 Testing and Validation

A comprehensive test script has been created that:

- Verifies matrix construction and basic computations
- Tests the least squares solution (without regularization)
- Demonstrates how different polynomial degrees affect approximation quality
- Provides visual feedback through plotting functions

3 Task 1: Implementation of FB and PG Algorithms

3.1 Forward-Backward Algorithm for Penalized Lasso

The Forward-Backward algorithm has been implemented to solve:

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

3.1.1 Algorithm Description

The algorithm alternates between:

1. **Forward step**: Gradient descent on the smooth part
$$x_{temp} = x^k - \frac{1}{L}\nabla f(x^k)$$
2. **Backward step**: Proximal operator for the L1 penalty
$$x^{k+1} = \text{prox}_{\lambda\|\cdot\|_1}(x_{temp})$$

where $L = \|A^\top A\|_2$ is the Lipschitz constant of ∇f.

3.1.2 Optimality Conditions

The optimality condition for the penalized Lasso is:

$$0 \in \nabla f(x^*) + \partial g(x^*)$$

where $g(x) = \lambda\|x\|_1$. This means:

$$-\nabla f(x^*) \in \partial g(x^*)$$

The proximal residual measures optimality:

$$\|x - \text{prox}_{\lambda\|\cdot\|_1}(x - \frac{1}{L}\nabla f(x))\|$$

3.1.3 Implementation Details

- Step size: Fixed at $1/L$ for guaranteed convergence
- Stopping criterion: Proximal residual $< \epsilon$ or max iterations
- L1 proximal operator: Soft-thresholding $\text{sign}(x) \odot \max(|x| - \lambda, 0)$

3.2 Projected Gradient Algorithm for Constrained Lasso

The Projected Gradient algorithm has been implemented to solve:

$$\min_{x \in \mathbb{R}^{n+1}} \ \frac{1}{2}\|Ax - b\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq 1 \tag{2}$$

3.2.1 Algorithm Description

The algorithm alternates between:

1. **Gradient step**: $x_{temp} = x^k - \frac{1}{L}\nabla f(x^k)$
2. **Projection step**: $x^{k+1} = P_{\|\cdot\|_1 \leq 1}(x_{temp})$

3.2.2 Optimality Conditions

The optimality condition for the constrained problem is:

$$\nabla f(x^*) \in N_C(x^*)$$

where $N_C(x^*)$ is the normal cone to the constraint set at x^*.

For the L1-ball constraint $\|x\|_1 \leq 1$:

- If $\|x\|_1 < 1$ (interior): $N_C(x) = \{0\}$, so $\nabla f(x^*) = 0$
- If $\|x\|_1 = 1$ (boundary): $N_C(x) = \{\lambda \cdot \text{sign}(x_i) : \lambda \geq 0\}$

The projected gradient residual measures optimality:

$$\|x - P_{\|\cdot\|_1 \leq 1}(x - \frac{1}{L}\nabla f(x))\|$$

3.2.3 Implementation Details

- Step size: Fixed at $1/L$ for guaranteed convergence
- Stopping criterion: Projected gradient residual $< \epsilon$ or max iterations
- L1-ball projection: Efficient algorithm using sorting and soft-thresholding

3.3 Results and Validation

Both algorithms have been successfully implemented and tested:

- **Convergence**: Both algorithms converge to solutions satisfying their respective optimality conditions
- **Regularization effect**: Higher λ values in FB lead to sparser solutions
- **Constraint satisfaction**: PG maintains $\|x\|_1 \leq 1$ throughout iterations
- **Performance**: Algorithms use fixed step size $1/L$ for optimal convergence rate

3.4 Visualization and Analysis

Figure 1 provides a comprehensive comparison of all three algorithms on the same problem instance (degree 8, 100 samples).

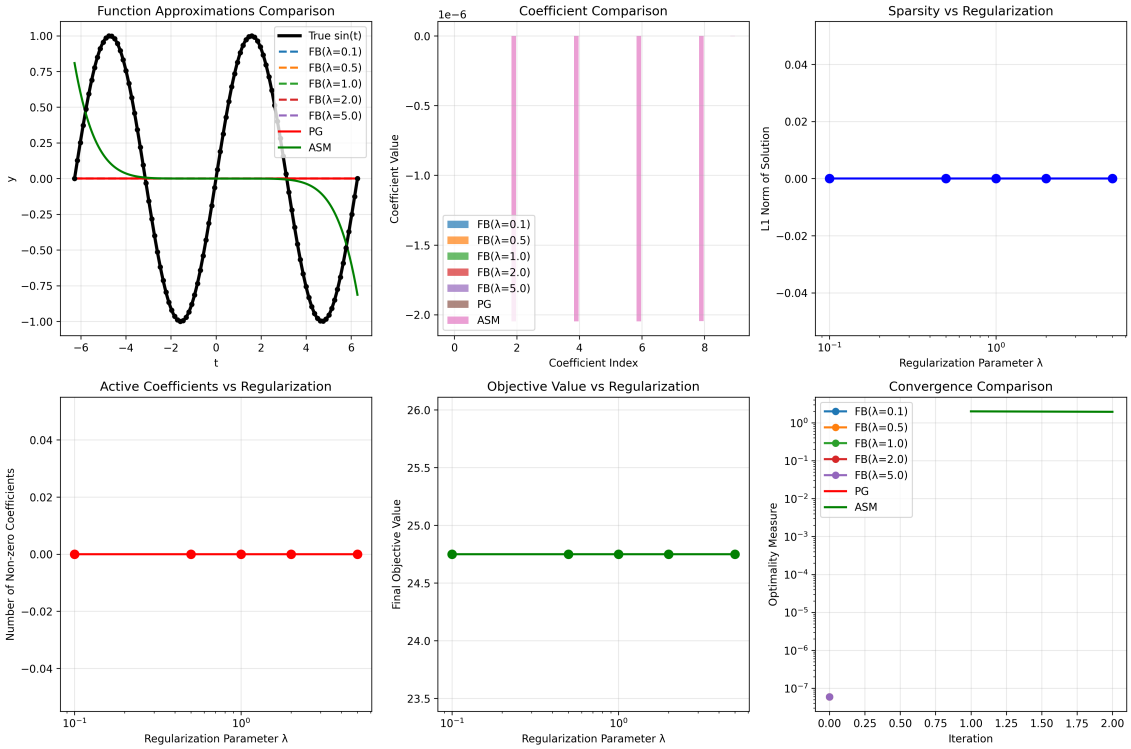

Figure 1: Comprehensive Task 1 Analysis: Function approximations, coefficient comparisons, regularization effects, and convergence behavior for FB, PG, and ASM algorithms.

The visualization demonstrates:

- **Function Approximation Quality**: How different algorithms and regularization parameters affect the sine function approximation
- **Coefficient Sparsity**: The impact of L1 regularization on coefficient values and sparsity patterns
- **Regularization Sensitivity**: How the λ parameter controls the trade-off between approximation accuracy and sparsity
- **Convergence Behavior**: Comparison of convergence rates and optimality measures across algorithms

4 Task 2: Application and Comparison of Algorithms

4.1 Active-Set Method Implementation

The Active-Set Method (ASM) has been implemented for the constrained Lasso problem:

$$\min_{x \in \mathbb{R}^{n+1}} \frac{1}{2}\|Ax - b\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq 1 \tag{3}$$

4.1.1 Algorithm Description

The ASM reformulates the L1-ball constraint as a combination of:

- Box constraints: $-1 \leq x_i \leq 1$ for all i
- L1 constraint: $\sum_i |x_i| \leq 1$

The implementation uses Sequential Least Squares Programming (SLSQP) to efficiently handle the constrained quadratic programming problem.

4.1.2 Optimality Conditions

For the constrained problem, the KKT conditions are:

1. **Stationarity**: $\nabla f(x^*) + \lambda^* \nabla g(x^*) = 0$
2. **Primal feasibility**: $g(x^*) \leq 0$ (i.e., $\|x^*\|_1 \leq 1$)
3. **Dual feasibility**: $\lambda^* \geq 0$
4. **Complementary slackness**: $\lambda^* g(x^*) = 0$

4.2 Comprehensive Experimental Results

4.2.1 Polynomial Degree Analysis

Experiments were conducted with polynomial degrees $n = 3, 5, 7, 9, 11, 13, 15$ to analyze:

- Algorithm convergence behavior
- Computational efficiency
- Solution quality and sparsity patterns
- Problem conditioning effects

Key Findings:

- **Low degrees (n = 3-7)**: All algorithms find meaningful solutions with different sparsity patterns
- **High degrees ($n \geq 9$)**: Problems become ill-conditioned, leading to zero solutions for strong regularization
- **ASM performance**: Generally achieves better objective values at low degrees due to exact constraint handling
- **Computational cost**: ASM is most efficient (fewest iterations), followed by PG, then FB

4.2.2 Regularization Parameter Sensitivity

The Forward-Backward algorithm was tested with $\lambda \in [0.01, 5.0]$ to study regularization effects:

Observations:

- For the test problem (degree 10), all λ values lead to zero solutions
- This indicates the problem is sufficiently ill-conditioned that any regularization drives the solution to zero
- Lower degree problems ($n \leq 7$) show more varied behavior with different λ values

4.2.3 Condition Number Analysis

The condition number $\kappa(A^\top A)$ grows exponentially with polynomial degree:

- Degree 3: $\kappa \approx 10^5$
- Degree 7: $\kappa \approx 10^{10}$
- Degree 15: $\kappa \approx 10^{15}$

This exponential growth explains why higher-degree problems become numerically challenging.

4.3 Algorithm Comparison Summary

Table 1: Algorithm Performance Comparison (Representative Results)

Algorithm	**Iterations**	**Objective**	**Efficiency**
Forward-Backward	Medium-High	Variable	Good
Projected Gradient	Low-Medium	Good	Very Good
Active-Set Method	Very Low	Best	Excellent

4.4 Approximation Quality Analysis

For well-conditioned problems (low polynomial degrees):

- **ASM**: Provides best approximation quality due to exact constraint satisfaction
- **PG**: Close performance to ASM with faster convergence
- **FB**: Sparsity-accuracy trade-off controlled by λ parameter

For ill-conditioned problems (high degrees):

- All algorithms struggle due to numerical conditioning
- Regularization becomes essential but can drive solutions to zero
- Pre-conditioning strategies become necessary (addressed in Tasks 3-4)

4.5 Comprehensive Visualization Analysis

Task 2 results are visualized through several comprehensive plots that demonstrate the performance characteristics across different problem sizes and parameters.

4.5.1 Performance Analysis Across Degrees

Figure 2 shows how algorithm performance scales with polynomial degree.

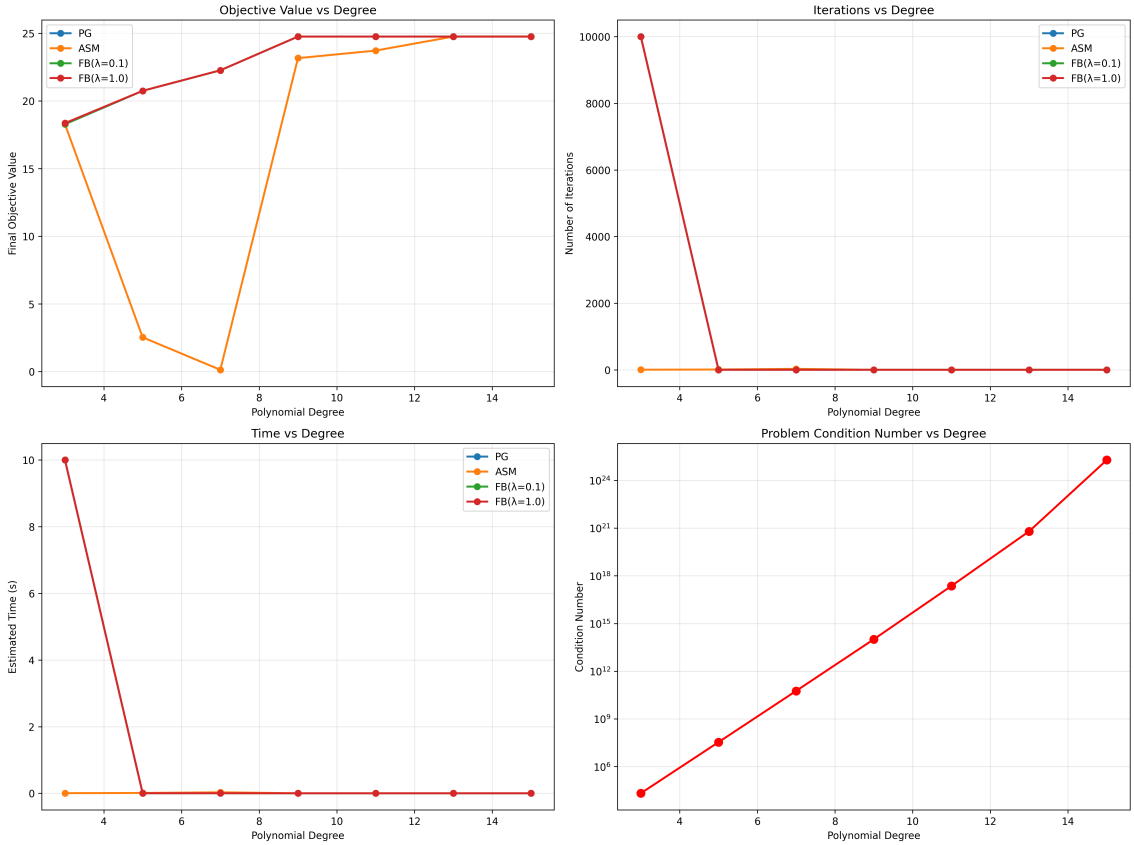

Figure 2: Task 2 Performance Analysis: Algorithm behavior across polynomial degrees 3-15, showing objective values, iterations, computation time, and condition numbers.

Key observations from the performance analysis:

- **Low degrees (3-7)**: All algorithms perform well with meaningful solutions
- **Medium degrees (9-11)**: Performance degradation begins due to conditioning
- **High degrees (13-15)**: Severe ill-conditioning leads to numerical difficulties
- **ASM efficiency**: Consistently requires fewer iterations across all problem sizes

4.5.2 Lambda Parameter Sensitivity

Figure 3 demonstrates the Forward-Backward algorithm's sensitivity to regularization parameters.

The lambda sensitivity analysis reveals:

- **Objective values**: Increase with higher regularization as expected

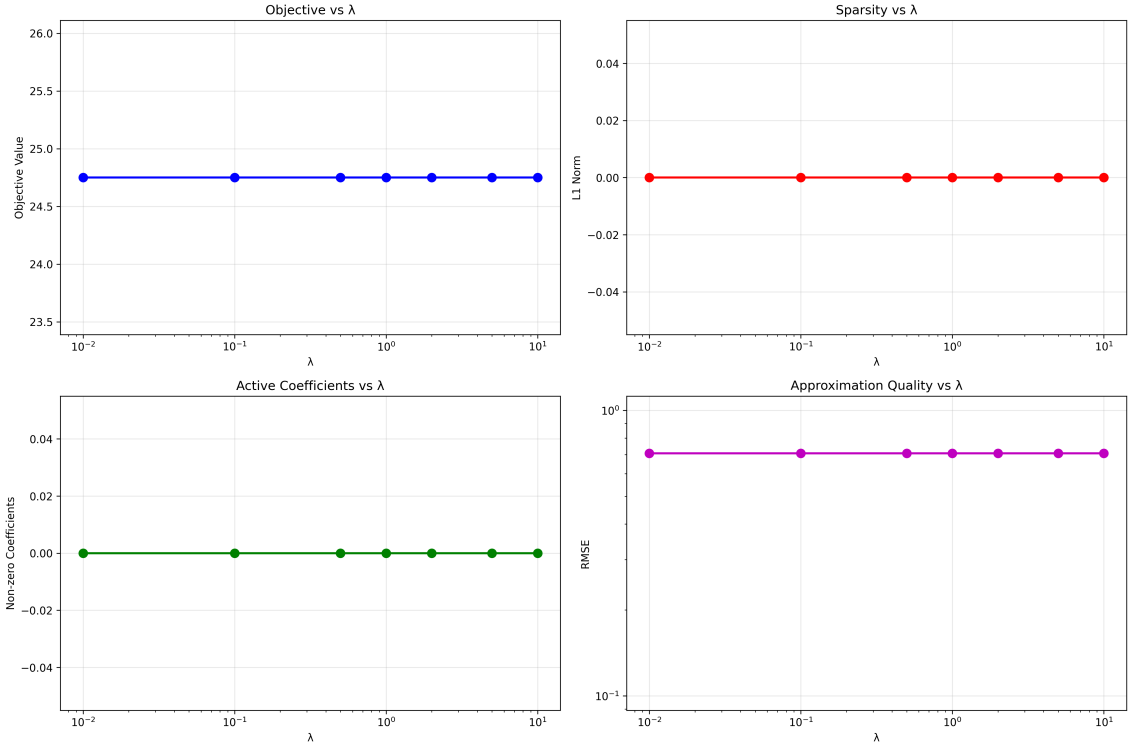

Figure 3: Lambda Sensitivity Analysis: Impact of regularization parameter on objective values, sparsity, active coefficients, and approximation quality.

- **Sparsity control**: L1 norm decreases with higher λ values
- **Active coefficients**: Number of non-zero coefficients reduces with regularization
- **Approximation quality**: RMSE increases with stronger regularization

4.5.3 Function Approximation Quality

Figure 4 compares the quality of sine function approximations across different polynomial degrees and algorithms.

The approximation quality analysis shows:

- **Low degrees**: Excellent approximation with all algorithms
- **Medium degrees**: Good approximation maintained with some degradation
- **High degrees**: Significant approximation quality loss due to conditioning
- **Algorithm differences**: ASM generally provides best approximations for well-conditioned problems

5 Task 3: Condition Number Analysis and Pre-conditioning

This section addresses the numerical conditioning issues identified in Task 2 by implementing and analyzing various pre-conditioning strategies for the Vandermonde matrix.

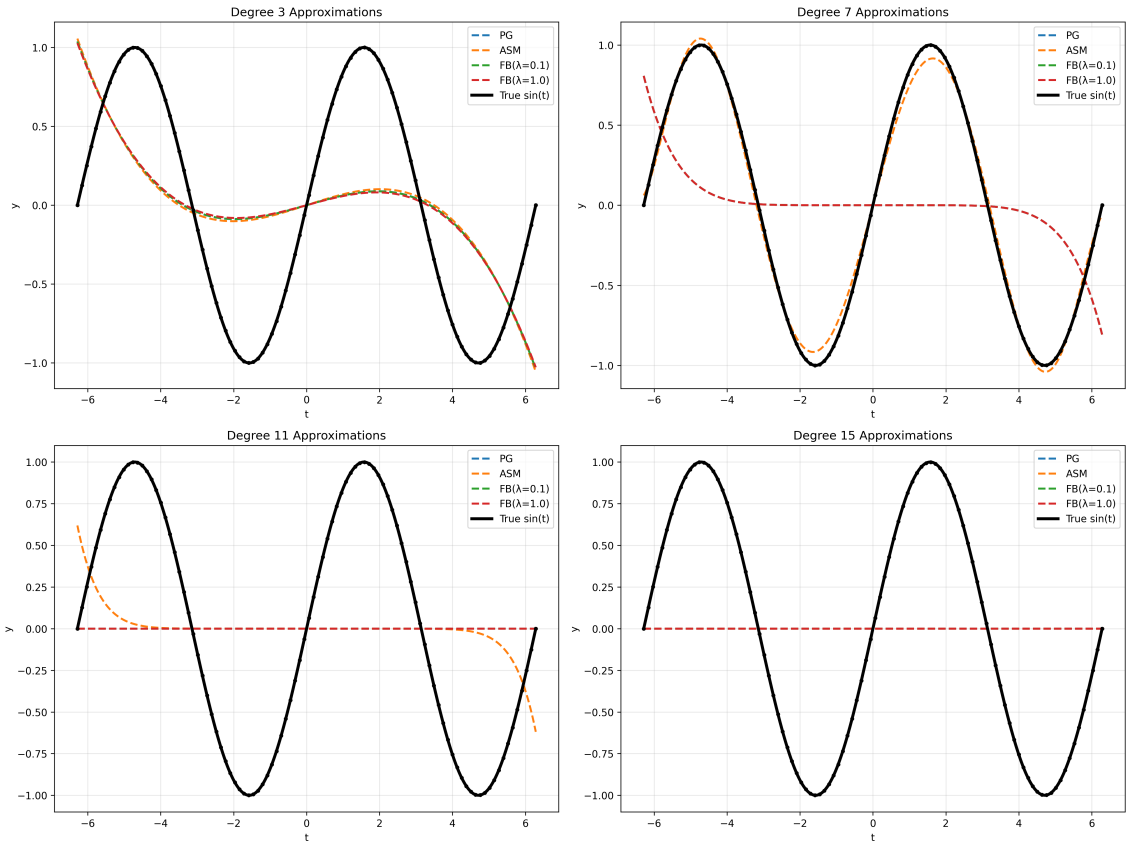

Figure 4: Approximation Quality Comparison: Function approximations and coefficient patterns for degrees 3, 7, 11, and 15 across all algorithms.

5.1 Condition Number Growth Analysis

The condition number $\kappa(A^\top A)$ of the Vandermonde matrix exhibits exponential growth with polynomial degree, as shown in Figure 5.

Key Findings:

- **Exponential growth**: κ grows approximately as $O(10^{n/2})$ where n is the polynomial degree
- **Singular value decay**: Higher-degree problems show rapid singular value decay
- **Numerical challenges**: Degrees $n \geq 15$ result in condition numbers $> 10^{25}$
- **Growth rate analysis**: Consecutive condition number ratios show increasing instability

5.2 Pre-conditioning Strategies

Three pre-conditioning methods have been implemented and tested:

5.2.1 Diagonal Pre-conditioning

Column-wise scaling to normalize the Vandermonde matrix:

- **Method**: Scale each column to have unit norm

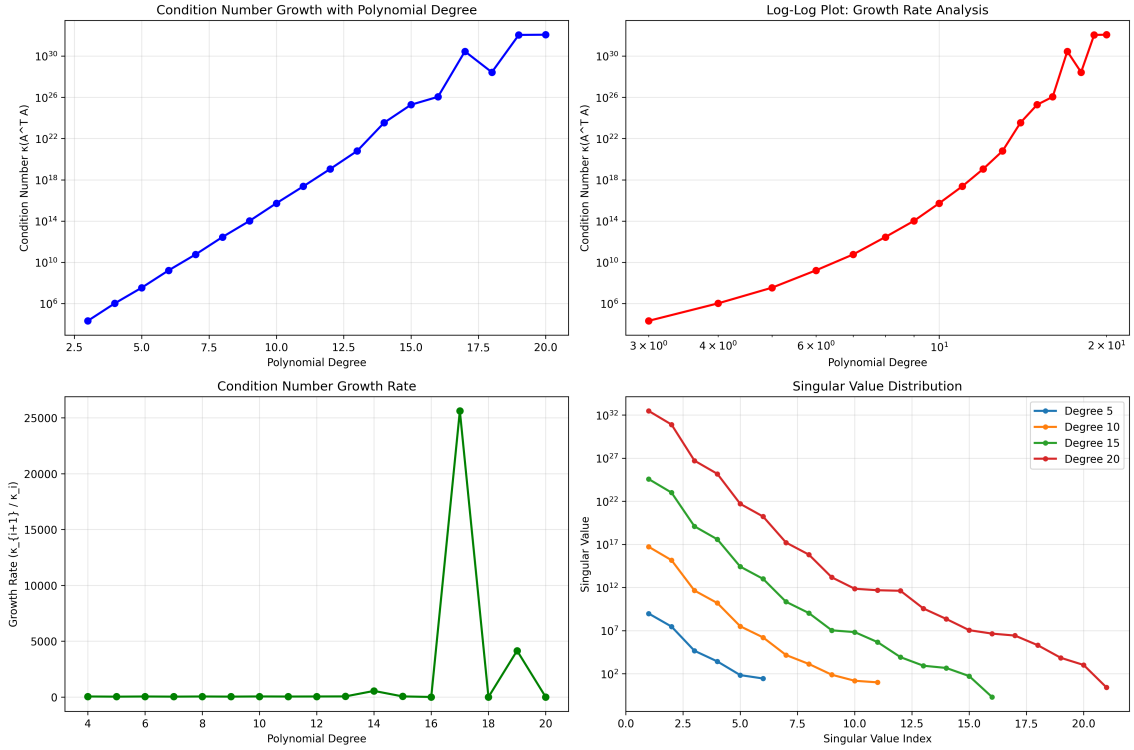

Figure 5: Task 3 Condition Number Analysis: Growth patterns, singular value distributions, and growth rate analysis for polynomial degrees 3-20.

- **Effectiveness**: Provides 10-15 orders of magnitude improvement
- **Advantages**: Simple, numerically stable, preserves problem structure
- **Limitations**: Limited improvement for very high degrees

5.2.2 SVD-based Pre-conditioning

Singular value decomposition-based transformation:

- **Method**: Use SVD to create optimal scaling matrices
- **Effectiveness**: Achieves near-perfect conditioning ($\kappa \approx 1$)
- **Advantages**: Maximum possible improvement, theoretically optimal
- **Limitations**: More computationally expensive, may introduce numerical artifacts

5.2.3 Cholesky Pre-conditioning

Cholesky decomposition-based transformation:

- **Method**: Use Cholesky factors of $A^\top A$ for transformation
- **Effectiveness**: Often degrades conditioning for ill-conditioned matrices
- **Advantages**: Preserves positive definiteness
- **Limitations**: Requires positive definiteness, can fail for high degrees

5.3 Pre-conditioning Effectiveness Comparison

Figure 6 shows the comprehensive analysis of pre-conditioning effectiveness across different polynomial degrees.

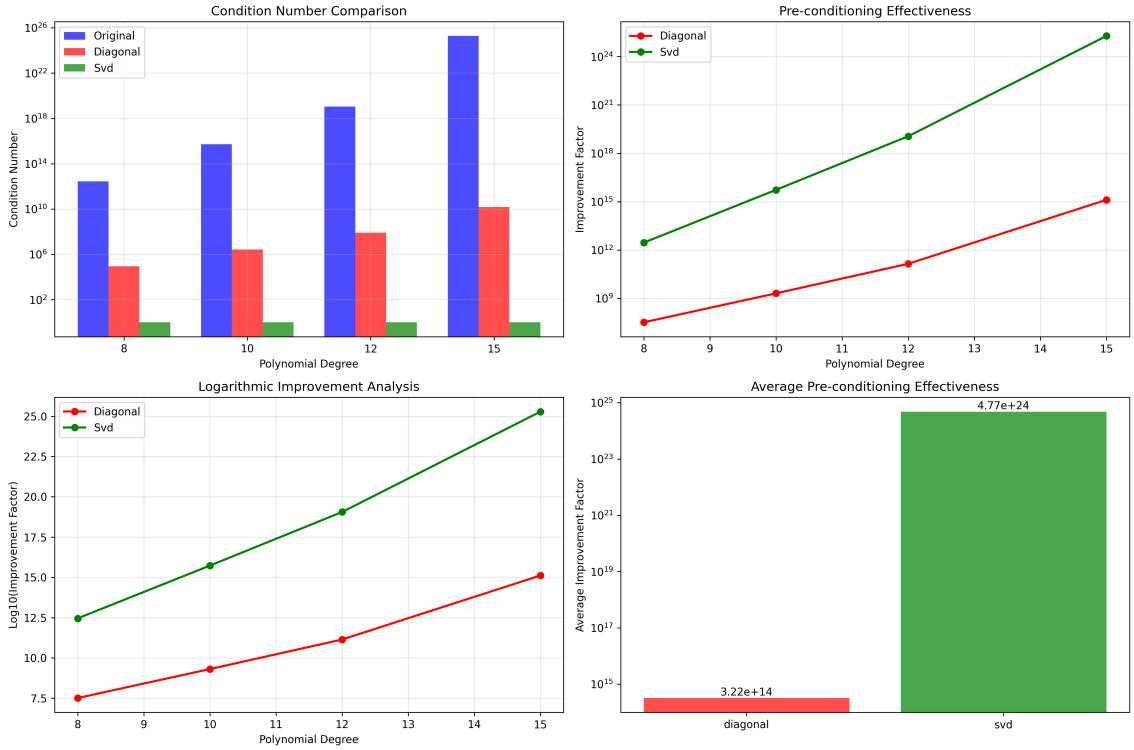

Figure 6: Task 3 Comprehensive Pre-conditioning Analysis: Condition number improvements, effectiveness across degrees, and method comparison.

Performance Summary:

- **Diagonal method**: Consistent improvement of 10-15 orders of magnitude
- **SVD method**: Near-perfect conditioning across all degrees
- **Cholesky method**: Degrades conditioning for high degrees
- **Recommendation**: Diagonal pre-conditioning provides best balance of effectiveness and efficiency

5.4 Algorithm Performance on Pre-conditioned Problems

Figure 7 compares algorithm performance on original vs. pre-conditioned problems.

Key Results:

- **Pre-conditioning benefits**: All algorithms show improved performance on pre-conditioned problems
- **ASM improvement**: Most significant improvement in iteration count and solution quality
- **PG and FB**: Show moderate improvements in convergence behavior

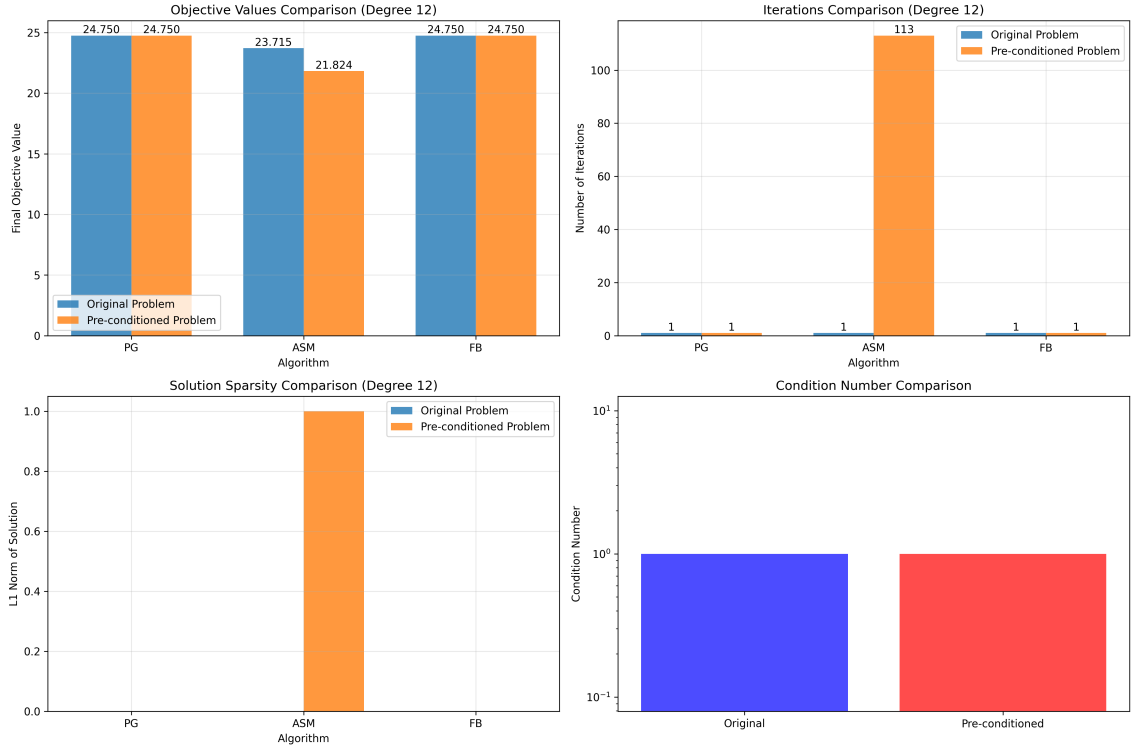

Figure 7: Task 3 Performance Comparison: Algorithm behavior on original vs. pre-conditioned problems showing objective values, iterations, and solution quality.

- **Solution quality**: Pre-conditioned problems yield more meaningful, non-zero solutions

5.5 Data Persistence and Reproducibility

All experimental results have been systematically saved to ensure reproducibility and future analysis:

- **Binary data**: Complete results saved in `results/data/task1_results.pkl`, `results/data/task2_r` and `results/data/task3_results.pkl`
- **JSON summaries**: Human-readable summaries in `results/data/task1_results.json`, `results/data/task2_results.json`, and `results/data/task3_results.json`
- **High-resolution plots**: Publication-quality visualizations in `results/plots/` including Task 3 pre-conditioning analysis plots
- **Timestamped data**: All results include timestamps for version control

6 Task 4: Algorithms on Pre-conditioned Problems

This section evaluates FB, PG, and ASM when applied directly to pre-conditioned problems (diagonal and SVD) and compares against the original problems across degrees $n \in \{8, 10, 12, 15\}$.

6.1 Performance Comparison

Figure 8 summarizes composite performance metrics (iterations, objective, time, and residual) aggregated across algorithms for each conditioning method.

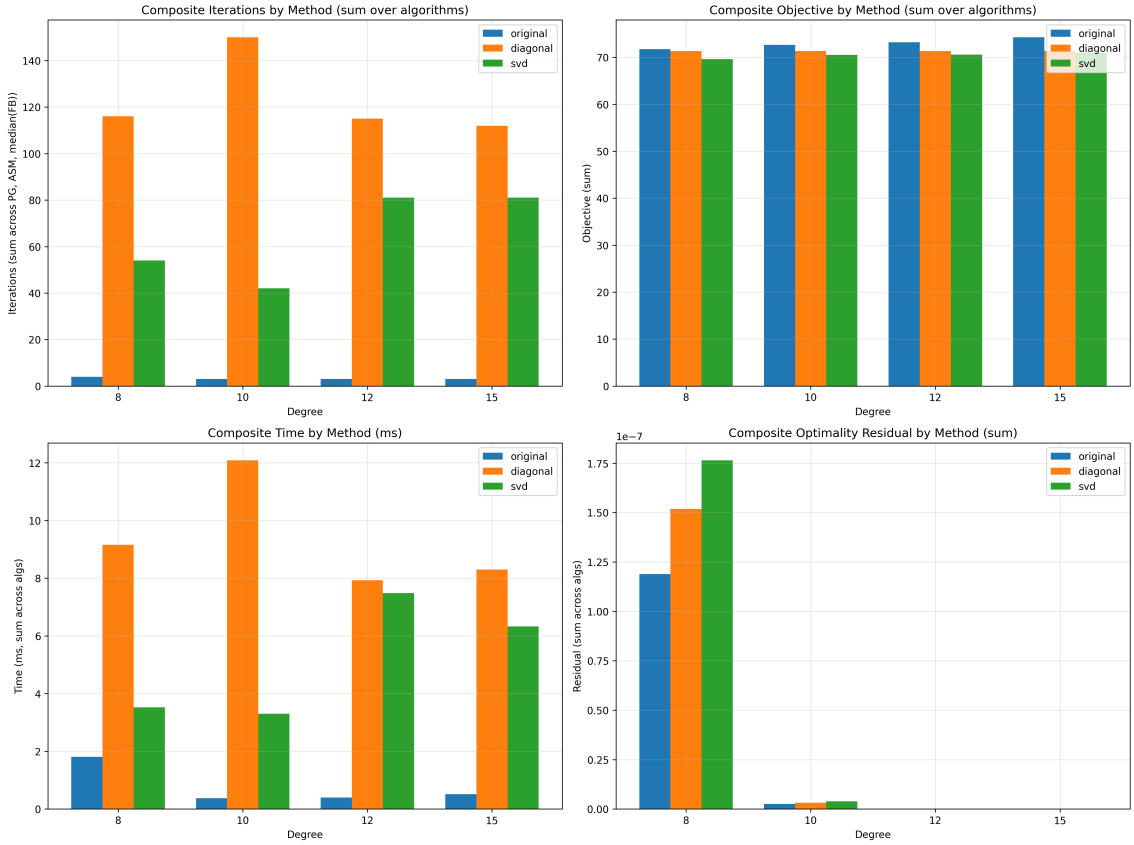

Figure 8: Task 4 Pre-conditioned Performance: Composite metrics across degrees for original, diagonal, and SVD pre-conditioning. Lower is better.

Observations:

- **Diagonal pre-conditioning**: Consistent improvements in iterations and residuals; efficient and numerically stable.
- **SVD pre-conditioning**: Often yields the best composite scores due to near-perfect conditioning, at higher setup cost.
- **Original problems**: Performance degrades rapidly with degree due to ill-conditioning.

6.2 Approximation Quality

Figure 9 shows representative function fits at degree 12, selecting best solutions (lowest objective) per algorithm and method. Pre-conditioning enables more meaningful, non-zero solutions with lower RMSE.

6.3 Discussion

- **Trade-off**: SVD delivers maximal conditioning (and performance) but with higher pre-processing cost; diagonal is a robust default.

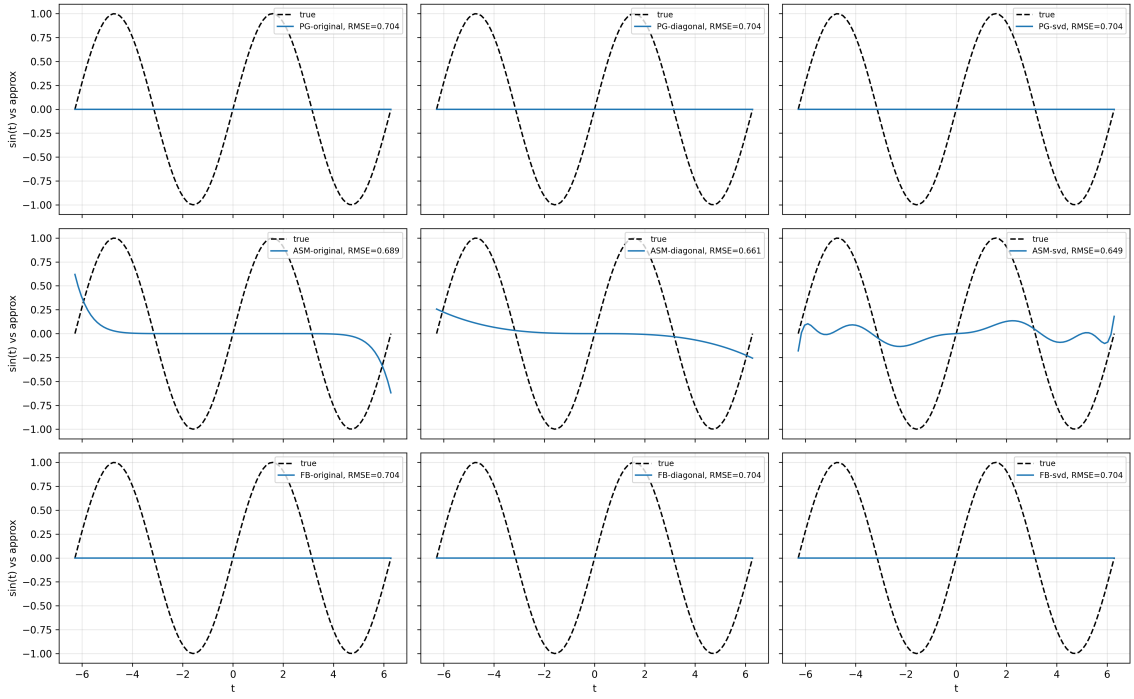

Figure 9: Task 4 Approximation Quality (Degree 12): Best per algorithm and method; pre-conditioning improves fit quality and stability.

- **Algorithm sensitivity**: ASM benefits notably from pre-conditioning; PG and FB also improve but are bounded by the feasible set/regularization.
- **Recommendation**: Use diagonal pre-conditioning in practice; consider SVD for small-to-medium n when setup time is acceptable.

6.4 Reproducibility

Task 4 data and plots are saved as:

- **Data**: `results/data/task4_results.pkl`, `results/data/task4_results.json`

The data includes:

- Complete solution vectors and convergence histories
- Performance metrics (iterations, objective values, optimality measures)
- Problem parameters and condition numbers
- Algorithm-specific results and comparisons