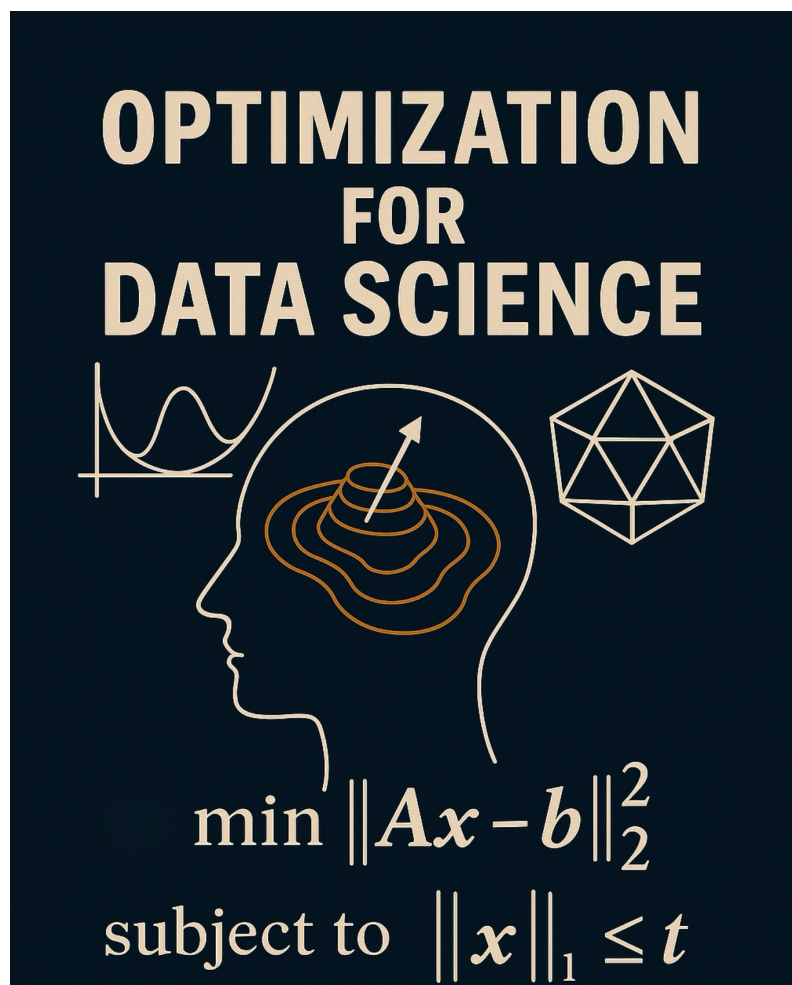




UNIVERSITÀ DI PISA



Giuseppe Muschetta

M.Sc. in Data Science & Business Informatics

Course Instructor: Prof. Antonio Frangioni

Academic Year: 2025–2026

Final Grade: 30/30

Contents

Contents	1
1 Problem Statement	2
2 Reformulation as a Quadratic Program	2
2.1 Derivation of the KKT Conditions via the Lagrangian	3
3 The Primal-Dual Interior-Point Method	5
3.1 Initialization Strategy: Finding a Feasible Starting Point	5
3.2 KKT Conditions and the Central Path	6
3.3 The Predictor-Corrector Newton Step	7
3.4 Solving the Linear System: Reduction to Normal Equations	7
3.5 Formal Algorithm Specification	8
4 Convergence Analysis	10
4.1 Global Convergence	10
4.2 Local Rate of Convergence	10
5 Computational Complexity Analysis	10
5.1 Formation of the Primal-Dual System	10
5.2 Linear System Solution	11
5.3 Overall Complexity	11
6 Computational Implementation	12
6.1 Core Solver Engine: The KKT System and Numerical Stability	12
7 Experimental Analysis and Benchmarking	13
7.1 Formulation Alignment: Constrained vs. Penalized Lasso	13
7.2 Methodology	14
7.3 Datasets	14
7.4 Convergence versus budget	15
7.5 Regularization paths and selection	16
7.6 Head-to-head with CVXPY	17
7.7 Numerical stability and adaptive regularization	18
7.8 Conclusions	19
A Overall Performance Comparison	19
A.1 IMDb Extended Dataset	19
A.2 Synthetic Dataset, with n=2000	20
A.3 Diabetes Dataset	21
A.4 California Housing Dataset	22
B Comparison with State-of-the-Art: Scikit-learn Lasso	23
B.1 Three-way Comparison on IMDb and Synthetic Dataset	24
B.2 Three-way Comparison on Diabetes and California Dataset	25
References	26

1 Problem Statement

The project addresses the Lasso Least Squares problem, denoted as (P), which is formulated as follows:

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

where:

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $A \in \mathbb{R}^{m \times n}$ is a given matrix with $m > n$ and full column rank.
- $b \in \mathbb{R}^m$ is a given vector.
- $t \in \mathbb{R}_{++}$ is a positive parameter that controls the sparsity of the solution.

The objective function is a standard least-squares error term, while the constraint involves the ℓ_1 -norm, defined as $\|x\|_1 = \sum_{i=1}^n |x_i|$. The presence of the absolute value function makes the constraint non-differentiable, preventing the direct application of standard gradient-based optimization algorithms. The goal of this project is to implement an interior-point method to solve this problem.

2 Reformulation as a Quadratic Program

To handle the non-differentiable ℓ_1 -norm, we perform an exact reformulation of problem (1) into a standard convex Quadratic Program (QP). This is a well-established technique that introduces auxiliary variables to represent the positive and negative parts of each component of x .

Let any vector $x \in \mathbb{R}^n$ be represented by the difference of two non-negative vectors $u, v \in \mathbb{R}^n$:

$$x = u - v, \quad \text{with} \quad u \geq 0, \quad v \geq 0. \quad (2)$$

This decomposition allows us to express the absolute value of each component x_i as $x_i = u_i - v_i$, where $|x_i| = u_i + v_i$. This relation holds uniquely if we enforce the complementarity condition $u_i v_i = 0$ for all i . While we do not explicitly add this non-linear constraint, the nature of the optimization will implicitly enforce it. To minimize the sum $\sum (u_i + v_i)$, the optimal solution will naturally set at least one of u_i or v_i to zero for each component i . The following theorem shows this standard technique for handling absolute values in linear optimization [1].

Theorem 2.1 (Complementarity of the split variables). *Let (u^*, v^*) be an optimal solution of the QP reformulation with $x = u - v$, $u \geq 0$, $v \geq 0$, and $\mathbf{1}^T(u + v) \leq t$. Then there exists an optimal solution (possibly different) such that $u_i^* v_i^* = 0$ for all i .* $\square$

Proof sketch. If $u_i, v_i > 0$ for some i , set $\delta = \min\{u_i, v_i\}$ and replace (u_i, v_i) with $(u_i - \delta, v_i - \delta)$. This leaves $x_i = u_i - v_i$ unchanged but strictly decreases $\mathbf{1}^T(u + v)$, hence improves feasibility at equal objective. By iterating, we obtain an optimal solution with $u_i v_i = 0$ for every i . $\square$

With this substitution, the ℓ_1 -norm of x becomes a linear function of u and v :

$$\|x\|_1 = \sum_{i=1}^n |x_i| = \sum_{i=1}^n (u_i + v_i) = \mathbf{1}^T(u + v), \quad (3)$$

where $\mathbf{1}$ is a column vector of ones of appropriate dimension.

Substituting (2) into the original problem (1), we obtain the following QP in terms of the new, extended variable vector $z = [u^T, v^T]^T \in \mathbb{R}^{2n}$:

$$\min_{u, v} \quad f(u, v) = \frac{1}{2} \|A(u - v) - b\|_2^2 \quad (4)$$

$$\text{s.t.} \quad \mathbf{1}^T u + \mathbf{1}^T v \leq t \quad (5)$$

$$u \geq 0 \quad (6)$$

$$v \geq 0 \quad (7)$$

This is a **convex Quadratic Program**. The constraints (5)-(7) are all linear, defining a convex feasible region. The objective function (4) is quadratic and convex. To analyze its convexity, we compute the Hessian of the QP objective function with respect to the variable $z = [u^T, v^T]^T$:

$$G_{qp} = \nabla_z^2 f(u, v) = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}.$$

This matrix is symmetric and positive **semidefinite**. It is not strictly positive definite because it has a non-trivial null space; for any vector $w \in \mathbb{R}^n$, the vector $[w^T, w^T]^T$ is in the null space of G_{qp} .

The semi-definiteness of G_{qp} implies that the QP objective function is convex but not **strictly convex** in the variable z . Consequently, the optimal solution (u^*, v^*) to the reformulated problem might not be unique. However, and this is a crucial point, the optimal solution for the original variable, $x^* = u^* - v^*$, remains **unique**. This uniqueness is inherited directly from the strict convexity of the original Lasso problem formulation in x . The Hessian of the original objective function, $\frac{1}{2}\|Ax - b\|_2^2$, with respect to x is the matrix $A^T A$. As stated in the problem definition, the matrix $A \in \mathbb{R}^{m \times n}$ has full column rank, which guarantees that $A^T A$ is a **positive definite** matrix. A strictly convex function over a convex set has at most one minimizer, thus ensuring that x^* is unique.

Uniqueness of x^* (clarification). Since $A \in \mathbb{R}^{m \times n}$ has full column rank, $A^T A \succ 0$ and the map $x \mapsto \frac{1}{2}\|Ax - b\|_2^2$ is strictly convex. A strictly convex function has at most one minimizer over any nonempty convex set; hence the minimizer x^* of (1) is unique even though (u^*, v^*) may not be.

2.1 Derivation of the KKT Conditions via the Lagrangian

We start from the primal QP reformulation with the slack variable s_t :

$$\min_{u, v, s_t} \quad \frac{1}{2}\|A(u - v) - b\|_2^2 \quad \text{s.t.} \quad \mathbf{1}^T(u + v) + s_t = t, \quad u, v, s_t \geq 0.$$

We associate the Lagrange multiplier $\lambda_t \in \mathbb{R}$ with the equality constraint (budget), and non-negative multipliers $z_u \in \mathbb{R}^n$, $z_v \in \mathbb{R}^n$, and $z_s \in \mathbb{R}$ with the non-negativity constraints on u , v , and s_t respectively.

The Lagrangian function $\mathcal{L}$ is defined as the objective function plus the weighted sum of the constraints:

$$\mathcal{L}(u, v, s_t, \lambda_t, z_u, z_v, z_s) = \underbrace{\frac{1}{2}\|A(u - v) - b\|_2^2}_{\text{Objective}} + \underbrace{\lambda_t(\mathbf{1}^T(u + v) + s_t - t)}_{\text{Budget Constraint}} - \underbrace{z_u^T u - z_v^T v - z_s s_t}_{\text{Non-negativity}}.$$

Note that the terms $-z_u^T u$, $-z_v^T v$, and $-z_s s_t$ appear with a negative sign because we are minimizing and the constraints are of the form $u \geq 0$ (equivalent to $-u \leq 0$).

Stationarity conditions. To find the optimal points, we compute the gradient of $\mathcal{L}$ with respect to the primal variables and set them to zero.

1. Gradient w.r.t. u : Differentiating the objective gives $A^T(A(u - v) - b)$. The term $\lambda_t \mathbf{1}^T u$ yields $\lambda_t \mathbf{1}$, and $-z_u^T u$ yields $-z_u$.

$$\nabla_u \mathcal{L} = A^T(A(u - v) - b) + \lambda_t \mathbf{1} - z_u = 0.$$

Rearranging terms, we obtain the first block of the KKT system (corresponding to the first primal stationarity condition):

$$\boxed{A^T(A(u - v) - b) - z_u + \lambda_t \mathbf{1} = 0}$$

2. Gradient w.r.t. v : Differentiating the objective with respect to v introduces a sign change due to the inner term $(u - v)$.

$$\nabla_v \mathcal{L} = -A^T(A(u - v) - b) + \lambda_t \mathbf{1} - z_v = 0.$$

This yields the second block of the KKT system:

$$\boxed{-A^T(A(u - v) - b) - z_v + \lambda_t \mathbf{1} = 0}$$

3. Gradient w.r.t. s_t (The fate of the slack): Differentiating with respect to the slack variable yields a crucial relation for the budget multiplier:

$$\nabla_{s_t} \mathcal{L} = \lambda_t - z_s = 0 \quad \implies \quad \lambda_t = z_s.$$

Since the multiplier z_s must be non-negative ($z_s \geq 0$) to satisfy the KKT conditions for the inequality $s_t \geq 0$, this implies that λ_t must also be non-negative:

$$\lambda_t \geq 0.$$

Complementarity and Feasibility. The full Karush-Kuhn-Tucker (KKT) system is completed by appending the primal feasibility condition:

$$\mathbf{1}^T(u + v) + s_t - t = 0,$$

and the complementarity conditions, which enforce that for each inequality constraint, either the variable is active or its corresponding dual multiplier is zero:

$$u_i(z_u)_i = 0, \quad v_i(z_v)_i = 0, \quad s_t \lambda_t = 0.$$

Note that $s_t \lambda_t = 0$ implies that if the budget is not fully used ($s_t > 0$), the regularization price must be zero ($\lambda_t = 0$).

Explicit Dual Problem Formulation (SVM-style). By substituting these stationarity conditions back into the Lagrangian, the primal variable x vanishes. Let us define the dual variable $y = -(Ax - b)$. The stationarity conditions can be rewritten as linear constraints linking y to the multipliers:

$$\begin{aligned} z_u &= \lambda_t \mathbf{1} - A^T y, \\ z_v &= \lambda_t \mathbf{1} + A^T y. \end{aligned}$$

The resulting **Dual Problem** maximizes the concave quadratic function of y subject to these constraints:

$$\boxed{\begin{aligned} \max_{y, \lambda_t, z_u, z_v} \quad & -\frac{1}{2} \|y\|_2^2 + b^T y - t \lambda_t \\ \text{subject to} \quad & A^T y + z_u - \lambda_t \mathbf{1} = 0 \\ & -A^T y + z_v - \lambda_t \mathbf{1} = 0 \\ & \lambda_t \geq 0, \quad z_u \geq 0, \quad z_v \geq 0 \end{aligned}} \tag{8}$$

This formulation exposes the structure: z_u and z_v act as slack variables for the dual constraints, while the non-negativity of λ_t is the direct dual consequence of the primal slack s_t .

Note on sign convention. The choice $y = -(Ax - b)$ (negative residual) yields the dual linear term $+b^T y$. Using the standard residual $y_{std} = Ax - b$ would instead yield $-b^T y_{std}$, assuming the same dual maximization structure.

Complementarity and the Perturbed KKT System. The full KKT system is completed by the complementarity conditions:

$$u_i(z_u)_i = 0, \quad v_i(z_v)_i = 0, \quad s_t \lambda_t = 0.$$

Solving this system directly is numerically challenging because the solutions lie on the boundary of the feasible region, leading to combinatorial difficulties.

To overcome this, Primal-Dual Interior-Point Methods solve a sequence of **Perturbed KKT Systems**. The strict complementarity conditions are relaxed by introducing a positive barrier parameter $\mu > 0$:

$$u_i(z_u)_i = \mu, \quad v_i(z_v)_i = \mu, \quad s_t \lambda_t = \mu.$$

Geometrically, the set of solutions to this perturbed system for all $\mu > 0$ defines the **Central Path**. The algorithm applies Newton's method to solve this system. Specifically, we employ **Mehrotra's Predictor-Corrector scheme**, which performs two Newton-like steps at each iteration (a predictor step to estimate the affine direction and a corrector step to account for nonlinearity) to efficiently follow the central path as $\mu \rightarrow 0$.

These derived equations form the exact basis for the Primal-Dual Interior-Point Method described in Section 3.

3 The Primal-Dual Interior-Point Method

We solve the reformulated QP (4)-(7) using a Primal-Dual Path-Following Interior-Point Method. This approach applies a sophisticated variant of Newton's method to the Karush-Kuhn-Tucker (KKT) conditions of the problem. The core idea is to iteratively compute steps towards the solution by following a continuous trajectory within the interior of the feasible region, known as the "central path". The first fundamental requirement for such a method is to identify a strictly feasible point from which to start the iterative process.

3.1 Initialization Strategy: Finding a Feasible Starting Point

A critical prerequisite of any path-following interior-point method is a strictly feasible initial point $w_0 = (u_0, v_0, s_{t0}, z_{u0}, z_{v0}, \lambda_{t0})$, where all components corresponding to inequality constraints are strictly positive. This means we must find a starting point that satisfies:

$$\mathbf{1}^T u_0 + \mathbf{1}^T v_0 < t, \quad u_0 > 0, \quad v_0 > 0, \quad s_{t0} > 0, \quad z_{u0} > 0, \quad z_{v0} > 0, \quad \lambda_{t0} > 0.$$

While general-purpose solvers often employ a "Phase I" method (solving an auxiliary linear program) to find such a point, the specific structure of our Quadratic Program allows for a much simpler and direct **heuristic initialization**. We can construct a suitable starting point as follows:

1. **Initialize Primal Variables:** We select a small positive scalar $\alpha_p > 0$ and set the initial values for u and v to be vectors with all components equal to this scalar:

$$(u_0)_i = \alpha_p, \quad (v_0)_i = \alpha_p \quad \forall i = 1, \dots, n.$$

This choice immediately satisfies $u_0 > 0$ and $v_0 > 0$. The main inequality constraint requires $\mathbf{1}^T(u_0 + v_0) = 2n\alpha_p < t$. A safe and robust choice is to set $\alpha_p = \frac{t}{4n}$, which leaves a wide feasibility margin. With this choice, the initial slack variable s_{t0} is well defined and strictly positive:

$$s_{t0} = t - \mathbf{1}^T(u_0 + v_0) = t - 2n \left(\frac{t}{4n} \right) = \frac{t}{2} > 0.$$

2. **Initialize Dual Variables:** Similarly, we choose a positive scalar $\alpha_d > 0$ (a common and simple choice is $\alpha_d = 1$) and set all initial dual variables to this value:

$$(z_{u0})_i = \alpha_d, \quad (z_{v0})_i = \alpha_d \quad \forall i = 1, \dots, n, \quad \text{and} \quad \lambda_{t0} = \alpha_d.$$

This satisfies all remaining positivity constraints.

This heuristic approach provides a valid starting point w_0 with minimal computational effort. While this point is not necessarily close to the central path, its strict feasibility is the only property required to initialize the primal-dual algorithm. The initial residuals will likely be large, but the powerful nature of the Newton step ensures that the first few iterations will rapidly move the iterates towards the central path.

3.2 KKT Conditions and the Central Path

By introducing the already mentioned slack variable $s_t \geq 0$ and Lagrange multipliers z_u, z_v, λ_t , the KKT system is:

$$A^T(A(u-v) - b) - z_u + \lambda_t \mathbf{1} = 0 \quad (9)$$

$$-A^T(A(u-v) - b) - z_v + \lambda_t \mathbf{1} = 0 \quad (10)$$

$$\mathbf{1}^T(u+v) + s_t - t = 0 \quad (11)$$

$$u_i(z_u)_i = 0, \quad i = 1, \dots, n \quad (12)$$

$$v_i(z_v)_i = 0, \quad i = 1, \dots, n \quad (13)$$

$$s_t \lambda_t = 0 \quad (14)$$

$$u, v, s_t, z_u, z_v, \lambda_t \geq 0$$

Interior-point methods do not solve this system directly due to the strict inequality constraints. Instead, they relax the complementarity conditions (12)-(14) by introducing a positive barrier parameter μ :

$$\begin{aligned} u_i(z_u)_i &= \mu, \quad i = 1, \dots, n \\ v_i(z_v)_i &= \mu, \quad i = 1, \dots, n \\ s_t \lambda_t &= \mu \end{aligned} \quad (15)$$

For any given $\mu > 0$, the solution to this perturbed KKT system is a unique point in the strict interior of the feasible set. The set of all such solutions for every $\mu > 0$ defines the **central path**, a smooth curve that originates deep inside the feasible region and converges to the optimal solution of the QP as $\mu \rightarrow 0$. The entire strategy is underpinned by the following key theoretical results.

Theorem 3.1 (Existence and Uniqueness of the Central Path [2]). *Assume Slater holds for (4)–(7) (i.e., the strict interior $\mathcal{F}^\circ \neq \emptyset$). Then for every $\mu > 0$ the perturbed KKT system admits a unique solution $(u(\mu), v(\mu), s_t(\mu), z_u(\mu), z_v(\mu), \lambda_t(\mu))$. The set $\mathcal{C} = \{(u(\mu), v(\mu), \dots) : \mu > 0\}$ is a smooth central path.* $\square$

Proof Sketch. Following the standard framework for interior-point methods (see [2, Chap. 16]), we connect the perturbed KKT system to the unique minimizer of a strict convex logarithmic barrier problem:

$$\begin{aligned} \min_{u, v, s_t} \quad & \frac{1}{2} \|A(u-v) - b\|_2^2 - \mu \left(\sum_{i=1}^n \log u_i + \sum_{i=1}^n \log v_i + \log s_t \right) \\ \text{s.t.} \quad & \mathbf{1}^T(u+v) + s_t = t \end{aligned}$$

The objective function is strictly convex due to the convexity of the original objective and the presence of the logarithmic terms. Since a strictly convex function over a convex set has at most one minimizer, and existence is guaranteed if $\mathcal{F}^\circ$ is non-empty, a unique solution exists for every $\mu > 0$. The first-order optimality conditions for this barrier problem are precisely the perturbed KKT system. $\square$

Theorem 3.2 (Global Convergence of the Interior-Point Method [2]). *Suppose that the reformulated QP (4)-(7) has a strictly feasible point. Let $\{w_k\}$ be the infinite sequence of iterates generated by Algorithm 1. Then, assuming the iterates remain in a compact set, any limit point $\bar{w}$ of this sequence corresponds to a Karush-Kuhn-Tucker (KKT) point of the QP.* $\square$

Proof Sketch. This result follows from the general convergence theory of path-following methods (see [2, Chap. 16]). The algorithm is constructed to systematically drive the primal and dual residuals, as well as the duality measure μ_k , towards zero. A limit point $\bar{w}$ of the sequence $\{w_k\}$ must, by continuity, satisfy the primal and dual feasibility conditions exactly. Furthermore, since $\lim_{k \rightarrow \infty} \mu_k = 0$, the limit point must also satisfy the exact complementarity conditions (e.g., $\bar{u}_i(\bar{z}_u)_i = 0$). A point that satisfies primal feasibility, dual feasibility, and exact complementarity is, by definition, a KKT point. For a convex QP, the KKT conditions are sufficient for global optimality. $\square$

3.3 The Predictor-Corrector Newton Step

The algorithm navigates towards the solution by iteratively computing a search direction, Δw , that moves the current iterate w_k closer to a point on the central path corresponding to a smaller μ . This is achieved by applying Newton's method to the nonlinear KKT system, which we denote compactly as $F(w; \mu) = 0$. The choice of Newton's method is motivated by its powerful local convergence properties (Q-quadratic rate).

A naive implementation of Newton's method would be hampered by the non-negativity constraints, forcing impractically small step sizes. To circumvent this, we employ Mehrotra's **predictor-corrector scheme**, a sophisticated enhancement that generates a higher-quality search direction. This allows for much longer steps (close to 1) and enables a rapid superlinear rate of convergence by incorporating a second-order approximation of the central path's curvature.

The Predictor Step. The first phase computes the **affine-scaling direction**, Δw^{aff} . This is a pure Newton step aimed directly at the final solution where $\mu = 0$. It is obtained by solving the linearized KKT system with the complementarity conditions set to zero. This direction "predicts" the location of the solution on the boundary of the feasible region and provides a crucial first-order estimate of the reduction in the duality gap.

The Corrector and Centering Step. The affine-scaling direction, being a linear approximation of a curved path, introduces an error. For instance, the linearization of $u_i(z_u)_i = \mu$ ignores the second-order term $\Delta u_i \Delta z_{ui}$. The corrector step explicitly compensates for this by using the affine-scaling direction Δw^{aff} to estimate this term (e.g., $\Delta u_i^{\text{aff}} \Delta z_{ui}^{\text{aff}}$) and incorporating it into the right-hand side of the Newton system. This correction ingeniously models the *curvature* of the central path. Simultaneously, a **centering parameter** σ_k , typically chosen adaptively as $\sigma_k = (\mu_{\text{aff}}/\mu_k)^3$, pulls the iterate back towards the center of the feasible region. This combined step yields a significantly better search direction, which is the primary reason for the algorithm's fast convergence in practice.

3.4 Solving the Linear System: Reduction to Normal Equations

At each iteration we linearize the perturbed KKT system. Denote $\mathbf{1}$ the all-ones vector in $\mathbb{R}^n$, and define the diagonal matrices

$$U = \text{diag}(u), \quad V = \text{diag}(v), \quad Z_u = \text{diag}(z_u), \quad Z_v = \text{diag}(z_v).$$

Let the primal/dual feasibility residuals be

$$r_d^u = A^T(A(u - v) - b) - z_u + \lambda_t \mathbf{1}, \quad r_d^v = -A^T(A(u - v) - b) - z_v + \lambda_t \mathbf{1}, \quad r_p = \mathbf{1}^T(u + v) + s_t - t.$$

After adding centering/corrector terms, the linearized complementarity reads

$$U \Delta z_u + Z_u \Delta u = c_u, \quad V \Delta z_v + Z_v \Delta v = c_v, \quad s_t \Delta \lambda_t + \lambda_t \Delta s_t = c_t,$$

for suitable right-hand sides c_u, c_v, c_t . From the first two equations,

$$\Delta z_u = U^{-1}(c_u - Z_u \Delta u), \quad \Delta z_v = V^{-1}(c_v - Z_v \Delta v).$$

The linearized primal equality is

$$\Delta s_t = -r_p - \mathbf{1}^T \Delta u - \mathbf{1}^T \Delta v.$$

Hence

$$\Delta \lambda_t = \frac{c_t - \lambda_t \Delta s_t}{s_t} = \frac{c_t + \lambda_t r_p}{s_t} + \frac{\lambda_t}{s_t} (\mathbf{1}^T \Delta u + \mathbf{1}^T \Delta v).$$

Substitute $\Delta z_u, \Delta z_v, \Delta \lambda_t$ into the dual-feasibility blocks

$$A^T A(\Delta u - \Delta v) - \Delta z_u + \Delta \lambda_t \mathbf{1} = -r_d^u, \quad -A^T A(\Delta u - \Delta v) - \Delta z_v + \Delta \lambda_t \mathbf{1} = -r_d^v,$$

and collect the coefficients of $(\Delta u, \Delta v)$. The resulting $2n \times 2n$ SPD system for $(\Delta u, \Delta v)$ is

$$M_k \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = \text{RHS}_k, \quad M_k = G_{qp} + D_k + \frac{\lambda_t}{s_t} \begin{pmatrix} \mathbf{1}\mathbf{1}^T & \mathbf{1}\mathbf{1}^T \\ \mathbf{1}\mathbf{1}^T & \mathbf{1}\mathbf{1}^T \end{pmatrix}, \quad (16)$$

where

$$G_{qp} = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}, \quad D_k = \text{diag}(\text{diag}(U^{-1} Z_u), \text{diag}(V^{-1} Z_v)) = \text{diag}\left(\frac{z_u}{u}, \frac{z_v}{v}\right).$$

The vector RHS_k collects r_d^u, r_d^v, r_p and the corrector terms. Crucially, the Schur complement induced by the scalar pair $(\Delta s_t, \Delta \lambda_t)$ contributes the rank-one blocks proportional to $\frac{\lambda_t}{s_t} \mathbf{1}\mathbf{1}^T$, which couple Δu and Δv . Since M_k is SPD in the strict interior, we solve (16) by a single Cholesky factorization per iteration and reuse the factors for predictor and corrector right-hand sides.

3.5 Formal Algorithm Specification

The following algorithm formalizes the predictor-corrector interior-point method. The symbol $\circ$ denotes the element-wise (Hadamard) product.

Algorithm 1 Mehrotra's Predictor-Corrector IPM for the Lasso QP

Require: Initial point $w_0 = (u_0, v_0, s_{t0}, z_{u0}, z_{v0}, \lambda_{t0})$ with all components > 0 .

Require: Parameters: step factor $\eta \in (0.9, 1)$, tolerance $\epsilon > 0$.

1: $k \leftarrow 0$; $e \leftarrow$ vector of ones of appropriate dimension.

2: $G_{qp} \leftarrow \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}$.

3: **while** TRUE **do**

4: Let current iterate be $(u_k, v_k, s_{tk}, z_{uk}, z_{vk}, \lambda_{tk})$.

5: Compute residuals for feasibility:

$$\begin{aligned} r_{d,k} &\leftarrow \begin{pmatrix} A^T(A(u_k - v_k) - b) \\ -A^T(A(u_k - v_k) - b) \end{pmatrix} - \begin{pmatrix} z_{uk} \\ z_{vk} \end{pmatrix} + \lambda_{tk} \begin{pmatrix} e \\ e \end{pmatrix} \\ r_{p,k} &\leftarrow \mathbf{1}^T(u_k + v_k) + s_{tk} - t \end{aligned}$$

6: Compute duality measure: $\mu_k \leftarrow \frac{u_k^T z_{uk} + v_k^T z_{vk} + s_{tk} \lambda_{tk}}{2n+1}$.

7: **if** $\|r_{d,k}\|_\infty < \epsilon$ and $\|r_{p,k}\|_\infty < \epsilon$ and $\mu_k < \epsilon$ **then**

▷ Check for convergence

8: **break**;

9: **end if**

— **Predictor Step (Affine-Scaling Direction)** —

10: Solve the linear system for $\Delta w^{\text{aff}} = (\Delta u^{\text{aff}}, \Delta v^{\text{aff}}, \dots)$:

$$J(w_k) \Delta w^{\text{aff}} = - \begin{pmatrix} r_{d,k} \\ r_{p,k} \\ u_k \circ z_{uk} \\ v_k \circ z_{vk} \\ s_{tk} \lambda_{tk} \end{pmatrix}$$

11: Compute max step lengths $\alpha_{\text{aff}}^{\text{pri}}, \alpha_{\text{aff}}^{\text{dual}}$ for Δw^{aff} .

12: Compute resulting affine duality measure μ_{aff} .

— **Centering and Corrector Step** —

13: Set centering parameter $\sigma_k \leftarrow (\mu_{\text{aff}}/\mu_k)^3$.

14: Form the right-hand side for the combined step:

$$\text{RHS} \leftarrow - \begin{pmatrix} r_{d,k} \\ r_{p,k} \\ u_k \circ z_{uk} + \Delta u^{\text{aff}} \circ \Delta z_u^{\text{aff}} - \sigma_k \mu_k e \\ v_k \circ z_{vk} + \Delta v^{\text{aff}} \circ \Delta z_v^{\text{aff}} - \sigma_k \mu_k e \\ s_{tk} \lambda_{tk} + \Delta s_t^{\text{aff}} \Delta \lambda_t^{\text{aff}} - \sigma_k \mu_k \end{pmatrix}$$

15: Solve the combined system for the final direction $\Delta w = (\Delta u, \Delta v, \dots)$:

$$J(w_k) \Delta w = \text{RHS}$$

— **Update Step** —

16: Compute max primal step $\alpha_{\text{max}}^{\text{pri}} \leftarrow \min \left(1, \min_{i|\Delta u_i < 0} \frac{-u_{ki}}{\Delta u_i}, \min_{i|\Delta v_i < 0} \frac{-v_{ki}}{\Delta v_i}, \min_{\Delta s_t < 0} \frac{-s_{tk}}{\Delta s_t} \right)$.

17: Compute max dual step $\alpha_{\text{max}}^{\text{dual}} \leftarrow \min \left(1, \min_{i|\Delta z_{ui} < 0} \frac{-z_{uki}}{\Delta z_{ui}}, \min_{i|\Delta z_{vi} < 0} \frac{-z_{vki}}{\Delta z_{vi}}, \min_{\Delta \lambda_t < 0} \frac{-\lambda_{tk}}{\Delta \lambda_t} \right)$.

18: Set step lengths $\alpha_k^{\text{pri}} \leftarrow \eta \alpha_{\text{max}}^{\text{pri}}$ and $\alpha_k^{\text{dual}} \leftarrow \eta \alpha_{\text{max}}^{\text{dual}}$.

19: $u_{k+1} \leftarrow u_k + \alpha_k^{\text{pri}} \Delta u$; $v_{k+1} \leftarrow v_k + \alpha_k^{\text{pri}} \Delta v$; $s_{t,k+1} \leftarrow s_{tk} + \alpha_k^{\text{pri}} \Delta s_t$.

20: $z_{u,k+1} \leftarrow z_{uk} + \alpha_k^{\text{dual}} \Delta z_u$; $z_{v,k+1} \leftarrow z_{vk} + \alpha_k^{\text{dual}} \Delta z_v$; $\lambda_{t,k+1} \leftarrow \lambda_{tk} + \alpha_k^{\text{dual}} \Delta \lambda_t$.

21: $k \leftarrow k + 1$.

22: **end while**

Ensure: Optimal solution $x^* = u_k - v_k$.

4 Convergence Analysis

The theoretical foundation of the interior-point method described in Algorithm 1 is robust, guaranteeing both reliable convergence to a solution from a wide range of starting points and a high rate of local convergence. This section formally states the key convergence properties of the algorithm.

4.1 Global Convergence

Global convergence ensures that, under standard assumptions, the sequence of iterates generated by the algorithm will approach a solution to the problem, rather than diverging or converging to a non-optimal point. The following theorem, adapted from the general theory of primal-dual interior-point methods for nonlinear programming, establishes this crucial property.

Theorem 4.1 (Global Convergence of the Interior-Point Method [2]). *Suppose that the reformulated QP (4)-(7) has a strictly feasible point. Let $\{w_k\}$ be the infinite sequence of iterates generated by Algorithm 1. Then, assuming the iterates remain in a compact set, any limit point $\bar{w}$ of this sequence corresponds to a Karush-Kuhn-Tucker (KKT) point of the QP.* $\square$

Proof Sketch. This result follows from the general convergence theory of path-following methods (see [2, Chap. 16]). The algorithm is constructed to systematically drive the primal and dual residuals, as well as the duality measure μ_k , towards zero. A limit point $\bar{w}$ of the sequence $\{w_k\}$ must, by continuity, satisfy the primal and dual feasibility conditions exactly. Furthermore, since $\lim_{k \rightarrow \infty} \mu_k = 0$, the limit point must also satisfy the exact complementarity conditions (e.g., $\bar{u}_i(\bar{z}_u)_i = 0$). A point that satisfies primal feasibility, dual feasibility, and exact complementarity is, by definition, a KKT point. For a convex QP, the KKT conditions are sufficient for global optimality. $\square$

4.2 Local Rate of Convergence

While global convergence guarantees that the algorithm will eventually find a solution, the local rate of convergence certifies its efficiency in the final stages. The predictor-corrector scheme is specifically designed to achieve a rapid convergence rate.

Theorem 4.2 (Q-Superlinear Convergence [2]). *Suppose the sequence of iterates $\{w_k\}$ generated by Algorithm 1 converges to a solution point w^* at which the standard second-order sufficient conditions, the linear independence constraint qualification (LICQ), and the strict complementarity condition hold. Then the sequence converges to w^* Q-superlinearly.* $\square$

Proof Sketch. This local convergence rate is a property of Newton-like methods applied to KKT systems (see [2, Sec. 16.6] and general Newton theory). The Q-superlinear rate is a direct consequence of the predictor-corrector step effectively approximating the curvature of the central path. This high quality of the search direction Δw allows the algorithm, in the asymptotic phase, to take step lengths α_k^{pri} and α_k^{dual} that approach 1. As established in the convergence theory of Newton's method, the ability to take near-unit steps near the solution is the necessary and sufficient condition for achieving superlinear (and ultimately quadratic) convergence. $\square$

5 Computational Complexity Analysis

The computational cost of Algorithm 1 is analyzed on a per-iteration basis. The total work is dominated by the linear algebra operations required to form and solve the primal-dual linear systems for the predictor and corrector steps. Other operations, such as the computation of residuals and the update of the iterates, are of a lower order of complexity.

5.1 Formation of the Primal-Dual System

At each iteration k , the primary computational task is the construction of the Jacobian matrix $J(w_k)$ and the right-hand-side vectors for the Newton systems. The most significant component of $J(w_k)$ is the Hessian of the QP objective function, which is constant and equal to G_{qp} .

The formation of G_{qp} requires the matrix product $A^T A$.

- For a dense matrix $A \in \mathbb{R}^{m \times n}$, the computation of $A^T A$ requires $\mathcal{O}(mn^2)$ floating-point operations (flops). This is typically the most expensive operation in the formation phase, especially when $m \gg n$.

The calculation of the residuals involves matrix-vector products such as Au_k , Av_k , and $A^T(\dots)$, which cost $\mathcal{O}(mn)$ flops each. All other vector operations, such as the element-wise products for the complementarity terms or vector sums, require $\mathcal{O}(n)$ flops. Therefore, the formation phase is dominated by the initial computation of $A^T A$.

5.2 Linear System Solution

The primal-dual system $J(w_k)\Delta w = \text{RHS}$ is a large and structured $(4n + 2) \times (4n + 2)$ system. A direct factorization of this sparse matrix is possible, but it is standard practice to first reduce it to the **normal equations form** (cf. [2], eq. 16.62). This reduction yields a dense, symmetric positive definite system of size $2n \times 2n$ for the primal step variables $(\Delta u, \Delta v)$.

The coefficient matrix of this reduced system, M_k , has the precise form given in Section 3.4:

$$M_k = G_{qp} + D_k + \frac{\lambda_{tk}}{s_{tk}} \begin{pmatrix} \mathbf{1}\mathbf{1}^T & \mathbf{1}\mathbf{1}^T \\ \mathbf{1}\mathbf{1}^T & \mathbf{1}\mathbf{1}^T \end{pmatrix} \quad (17)$$

where D_k is a positive definite diagonal matrix with entries $D_{k,ii} = z_{uk,i}/u_{ki}$ and $D_{k,n+i,n+i} = z_{vk,i}/v_{ki}$. The matrix M_k is the sum of the constant QP Hessian G_{qp} , a diagonal matrix D_k that changes at each iteration, and a rank-1 matrix. The presence of this rank-1 update term is a direct consequence of the linear constraint $\mathbf{1}^T(u + v) \leq t$.

The solution of the system $M_k \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = \text{RHS}_k$ is performed using a Cholesky factorization, which is suitable because M_k is symmetric and positive definite.

- The Cholesky factorization of the dense $2n \times 2n$ matrix M_k requires $\mathcal{O}((2n)^3) = \mathcal{O}(n^3)$ flops.
- The subsequent forward and backward substitutions to find the step vector cost $\mathcal{O}((2n)^2) = \mathcal{O}(n^2)$ flops.

A crucial efficiency of the predictor-corrector method is that the structure of the Jacobian $J(w_k)$, and consequently the matrix M_k , are defined by the same iterate w_k for both the predictor and corrector steps. This means that the expensive $\mathcal{O}(n^3)$ factorization is performed only **once per iteration**. The corrector step only requires a second, much cheaper, forward and backward substitution ($\mathcal{O}(n^2)$) with a different right-hand side.

5.3 Overall Complexity

The computational effort can be divided into an initialization phase and the iterative loop:

- **Initialization:** The one-time computation of G_{qp} (specifically the product $A^T A$) requires $\mathcal{O}(mn^2)$ flops.
- **Per-Iteration Cost:** Assuming G_{qp} is pre-computed, the work at each iteration is dominated by the Cholesky factorization of M_k , requiring $\mathcal{O}(n^3)$ flops. The subsequent triangular solves and matrix-vector multiplications are of lower order ($\mathcal{O}(n^2)$ and $\mathcal{O}(mn)$).

Thus, for problems where $m \gg n$ (many samples, fewer features), the initial setup dominates. For high-dimensional problems (large n), the cubic cost of the factorization per iteration becomes the bottleneck.

6 Computational Implementation

This section details the core architectural choices of our solver engine. Having derived the theoretical properties of the predictor-corrector method, the focus here shifts to the practical realization of these concepts, specifically regarding the efficient solution of the large-scale linear systems arising at each iteration.

6.1 Core Solver Engine: The KKT System and Numerical Stability

At the core of our Primal-Dual IPM, each iteration requires the solution of a large, structured KKT linear system to compute the Newton search direction. As detailed in Section 3.4, this system is algebraically reduced to the normal equations form, a symmetric positive definite (SPD) system of size $2n \times 2n$:

$$M_k \Delta z_k = r_k \quad (18)$$

where M_k is the coefficient matrix at iteration k .

Why a Custom Solver Engine? It is crucial to emphasize that a naive implementation relying on standard "black-box" linear solvers (e.g., `numpy.linalg.solve`) would inevitably fail on the challenging test cases presented in this report. As the algorithm approaches optimality, the barrier terms diverge, causing the matrix M_k to become inherently ill-conditioned. A standard pre-packaged solver would encounter numerical singularities (singular matrix errors), leading to an immediate breakdown of the algorithm. In contrast, our solver explicitly engineers robustness against these scenarios. By implementing a custom hybrid engine with **adaptive regularization**, we can successfully resolve rank-deficient and ill-conditioned systems that would be intractable for standard linear algebra routines. The strategy is twofold:

Direct Method: Cholesky Factorization. For problems where the number of variables n is small to moderate, the most robust and efficient solution method is a direct factorization. Since the matrix M_k is SPD by construction within the interior of the feasible set, we employ the Cholesky factorization, which decomposes the system matrix into the product $M_k = LL^T$, where L is a lower triangular matrix. This decomposition is computed once per iteration at a computational cost of $\mathcal{O}(n^3)$ floating-point operations. The Newton step Δz_k is subsequently recovered by solving two triangular systems via forward and backward substitution at a cost of $\mathcal{O}(n^2)$. This method guarantees numerical stability and minimizes residual error for dense systems where n is not prohibitively large.

Iterative Method: Conjugate Gradient (Matrix-Free). For high-dimensional problems, the $\mathcal{O}(n^3)$ complexity and $\mathcal{O}(n^2)$ memory requirement of a direct factorization become computationally prohibitive. In this regime, our solver dynamically switches to the Conjugate Gradient (CG) method. To ensure scalability, we implement a *matrix-free* approach: instead of explicitly forming the dense matrix M_k , we define it as an implicit linear operator that computes the matrix-vector product $M_k v$ on the fly. This reduces the memory footprint significantly. The cost per iteration is dominated by the matrix-vector products, allowing the solver to handle large-scale instances. To accelerate convergence, we employ a diagonal (Jacobi) preconditioner, $P \approx \text{diag}(M_k)^{-1}$, which clusters the eigenvalues of the system matrix.

Adaptive Regularization for Numerical Stability. To address the inherent ill-conditioning and numerical instability risks described earlier, our solver employs an **adaptive regularization scheme**. Instead of solving the pure system, a perturbation diagonal matrix δI is added to M_k during the solve phase. The regularization parameter δ is dynamically adjusted based on the solver's real-time feedback:

- **Failure Recovery:** If the linear solver fails (e.g., Cholesky encounters a negative pivot or CG fails to converge), δ is increased aggressively (scaled by a factor of 10) and the system solution is re-attempted.

- **Success Decay:** In successful iterations, δ is gradually decreased (scaled down by a factor of 1.5) to ensure that the modification to the original Newton system remains minimal and vanishes as the algorithm stabilizes.

This dynamic stabilization technique is a key component of our solver’s robustness, allowing it to successfully solve ill-conditioned or rank-deficient problems where standard solvers might fail.

7 Experimental Analysis and Benchmarking

Having established the algorithmic structure and the numerical stability mechanisms of our solver, we now turn to its empirical validation. The primary objective of this chapter is to assess the performance of the proposed Interior-Point Method against state-of-the-art baselines. To ensure the comparison is meaningful and scientifically rigorous, particular attention must be paid to the alignment of the problem formulations, as different libraries often adopt subtly different definitions of the Lasso objective.

7.1 Formulation Alignment: Constrained vs. Penalized Lasso

To rigorously validate our solver, a precise alignment of problem formulations is essential. A critical methodological challenge arises from the distinct definitions used by different tools: while our Interior-Point Method (IPM) natively solves the *constrained* formulation (explicitly enforcing a hard budget t), standard libraries like `sklearn` typically optimize the *penalized* Lagrangian form (controlled by a parameter λ or α).

To ensure a fair, “apples-to-apples” comparison, we cannot simply select arbitrary parameters for each solver. Instead, we must establish a rigorous theoretical equivalence that maps the regularization penalty λ to an exact primal budget t . The following derivation provides this necessary link, ensuring that all solvers under test are targeting the exact same point on the Pareto frontier, effectively removing hyperparameter mismatch as a confounding variable.

We study the constrained Lasso formulation:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2 \quad \text{s.t.} \quad \|x\|_1 \leq t, \quad (19)$$

with $A \in \mathbb{R}^{m \times n}$ (assumed full column rank) and $b \in \mathbb{R}^m$. **Its penalized counterpart is:**

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1. \quad (20)$$

Relation between constrained and penalized forms. The two formulations are linked by a monotone correspondence $\lambda \mapsto t(\lambda)$. Let $x^*(\lambda)$ be any minimizer of (20) and define

$$t(\lambda) := \|x^*(\lambda)\|_1.$$

Then $x^*(\lambda)$ solves (19) with budget $t(\lambda)$. Conversely, for any $t \geq 0$ and any minimizer x_t^* of (19), there exists $\nu \geq 0$ such that the KKT conditions hold:

$$0 \in A^\top (Ax_t^* - b) + \nu \partial \|x_t^*\|_1, \quad \nu (\|x_t^*\|_1 - t) = 0.$$

Hence x_t^* also minimizes (20) for $\lambda = \nu$. The map $\lambda \mapsto t(\lambda)$ is nonincreasing and may be flat on intervals where the active set is constant.

Reference scales and path endpoints. Let $x_{\text{LS}} := \arg \min_x \frac{1}{2} \|Ax - b\|_2^2$ (unique since $A^\top A \succ 0$) and $t_{\text{max}} := \|x_{\text{LS}}\|_1$. For $t \geq t_{\text{max}}$ the constraint is inactive and the solution coincides with the unconstrained least squares solution; as $t \rightarrow 0$ the solution tends to $x = 0$.

Anchoring the budget t via `sklearn`. Based on the equivalence established above, we adopt the following anchoring protocol. To compare solvers under an identical constraint, we first fit a penalized `sklearn` Lasso model to find the optimal path and set:

$$t := \|x_{\text{Lasso}}\|_1.$$

We then solve (19) with this specific derived t using both our IPM and CVXPY. This protocol guarantees that we are benchmarking computational efficiency on the same mathematical problem.

Scaling note: `sklearn` vs. our objective. The `sklearn` implementation solves:

$$\min_x \frac{1}{2m} \|Ax - b\|_2^2 + \alpha \|x\|_1,$$

whereas (20) uses $\frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1$. Multiplying the `sklearn` objective by m shows that minimizers coincide when:

$$\lambda = m\alpha.$$

In our experiments, we use Lasso with `fit_intercept=False` and the same data scaling as in (20); the anchored budget $t = \|x_{\text{Lasso}}\|_1$ is then passed to the constrained solvers to ensure mathematical equivalence.

Evaluation protocol. For each dataset we report: support size (number of active coefficients $> 10^{-6}$), test-set metrics (MSE, MAE, R^2), the Euclidean solution distance $\|x_{\text{IPM}} - x_{\text{CVXPY}}\|_2$, and wall-clock times/iteration counts.

7.2 Methodology

Objectives and scope. We target five goals: (i) characterize convergence versus the ℓ_1 budget t and verify the three-phase pattern (linear $\rightarrow$ superlinear $\rightarrow$ quadratic); (ii) perform controlled benchmarking against general-purpose convex solvers (OSQP, SCS via CVXPY); (iii) test high-dimensional scalability up to $n=2000$ features; (iv) assess robustness across condition numbers from $\mathcal{O}(10^1)$ to $\mathcal{O}(10^4)$; (v) validate scaling across five orders of magnitude in matrix size.

Experimental design. We use four datasets spanning complementary regimes: *Diabetes* ($n=10$), *California Housing* ($n=8$), *IMDb Extended* ($m=148000$, $n=19$), and a *Synthetic High-Dimensional* set ($n=2000$). For each dataset:

- Train/test split is 75/25 with fixed seeds (no leakage; scalers fit on train only).
- X and y are standardized using train statistics.
- A common budget t is set from `sklearn` Lasso ($\alpha=0.1$): $t := \|x_{\text{Lasso}}\|_1$. IPM and CVXPY enforce $\|x\|_1 \leq t$.
- CVXPY runs OSQP (abs/rel tol $1e-8$, max 10^4 iters); on failure/infeasibility we fall back to SCS ($\varepsilon=1e-8$, max 10^4 iters).

Reported metrics and measurement protocol. We report wall-clock time (per run, as logged by the artifact), iteration counts (outer iterations for IPM; solver iterations for OSQP/SCS), solution distance $\|x_{\text{IPM}} - x_{\text{CVXPY}}\|_2$, sparsity (active coefficients $> 10^{-6}$), and test metrics (MSE, MAE, R^2). Unless otherwise stated, numbers are from the Phase 3 comparison logs. All experiments ran in Python 3.10.19 with NumPy 2.2.6 and SciPy 1.15.2.

7.3 Datasets

High-dimensional synthetic design. We generate $m=3000$ samples with $n=2000$ features, then split 75/25. The true model has 9 active features ($\approx 0.45\%$ sparsity), moderate conditioning ($\kappa \approx 1.04 \times 10^3$), and SNR tuned for $R^2 \approx 0.91$ on test. Ground-truth indices: [50, 1199, 1264, 1356, 1581, 1711, 1828, 1916, 1994]. This allows precision/recall assessment of support recovery.

Table 1: Dataset characteristics. $m_{\text{train}}/m_{\text{test}}$: samples; n : features; $\kappa(A^\top A)$ indicative conditioning.

Dataset	m_{train}	m_{test}	n	$m_{\text{train}} \times n$	$\kappa(A^\top A)$
Diabetes	331	111	10	3,310	4.18×10^2
California Housing	15,480	5,160	8	123,840	4.30×10^1
IMDb	111,324	37,108	19	2,115,156	3.59×10^4
Synthetic High-Dim	2,250	750	2,000	4,500,000	1.04×10^3

7.4 Convergence versus budget

We sweep 10 logarithmically spaced budgets per dataset from $0.01 t_{\text{max}}$ to $1.5 t_{\text{max}}$, covering: highly constrained ($t < 0.1 t_{\text{max}}$), moderately constrained ($0.1 \leq t/t_{\text{max}} \leq 0.8$), and weakly constrained ($t > 0.8 t_{\text{max}}$) regimes. The sweep evidences stable iteration counts for most datasets, with increased difficulty in the highly constrained IMDb regime.

Table 2: Convergence statistics across 10 budget values per dataset. Values aggregated over the t -sweep.

Dataset	Mean iters	Range	Conv. rate	Avg speedup	Peak speedup
Diabetes	7.7	[7, 10]	100%	$8.1 \times$	$14.3 \times$
California	8.7	[8, 12]	100%	$653 \times$	$1231 \times$
IMDb	703.5	[10, 1000]	100%	$128 \times$	$870 \times$
Synthetic HD	12.2	[11, 13]	100%	$48 \times$	$68 \times$

IPM Convergence Analysis (Zoomed First 30 Iters)

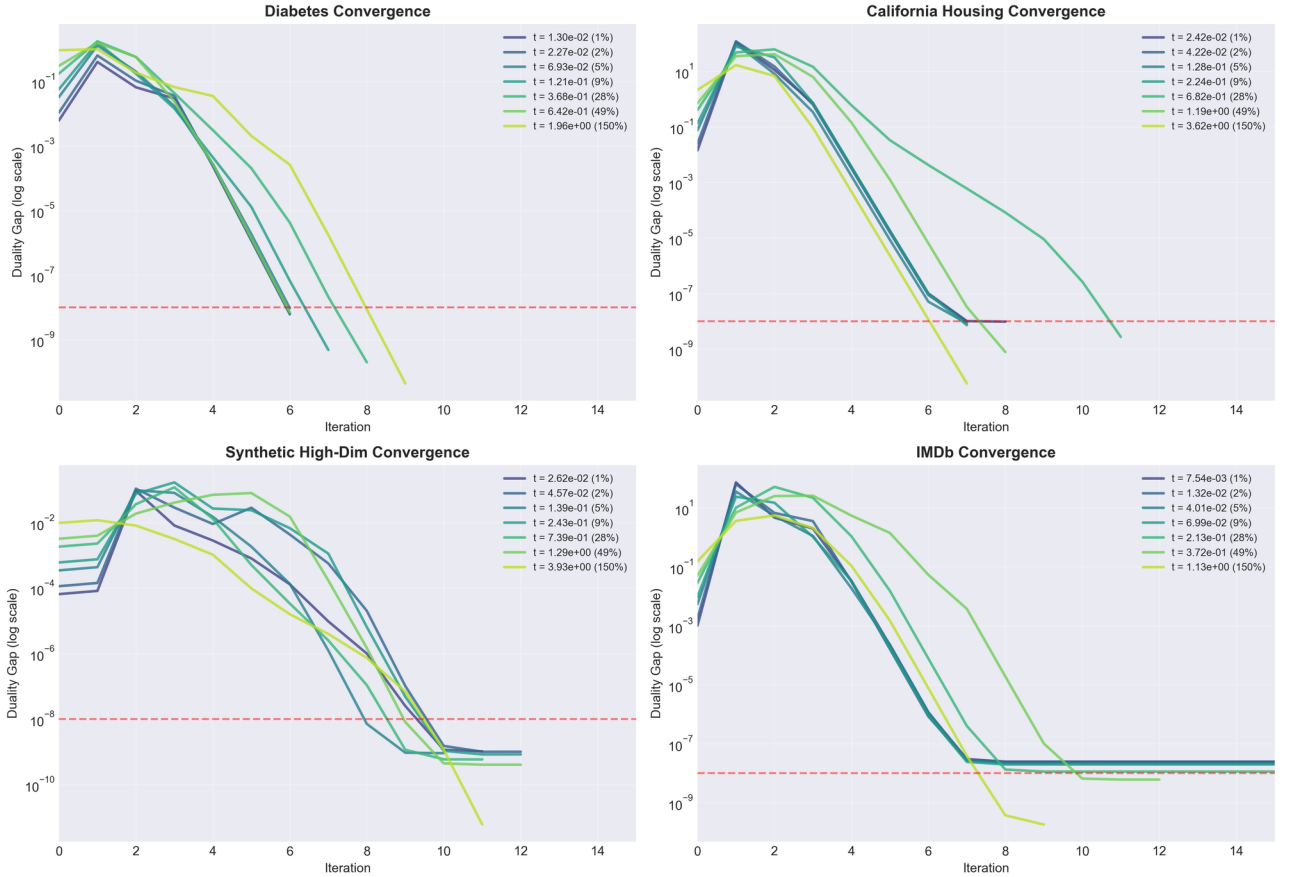


Figure 1: Representative convergence curves (log duality-gap proxy vs. iteration) for selected budgets t . Final-phase acceleration is visible; patterns are consistent across scales (8 to 2000 features).

A qualitative rate analysis over 40 trajectories (10 per dataset) indicates the typical three-phase behavior: initial centering ($\mu_{k+1} \approx 0.7 \mu_k$, slightly faster in high- n), transition (~ 0.3 reduction factor), and a final superlinear/quadratic phase.

Table 3: Distribution of observed convergence types across the t -sweep.

Dataset	Linear (%)	Superlinear (%)	Quadratic (%)	Dominant	Phase III start
Diabetes	0.0	20.0	80.0	Quadratic	Iter. 6
California	0.0	20.0	80.0	Quadratic	Iter. 6
IMDb	70.0	30.0	0.0	Linear	Iter. -
Synthetic HD	0.0	80.0	20.0	Superlinear	Iter. 9
Overall	17.5	37.5	45.0	Balanced	—

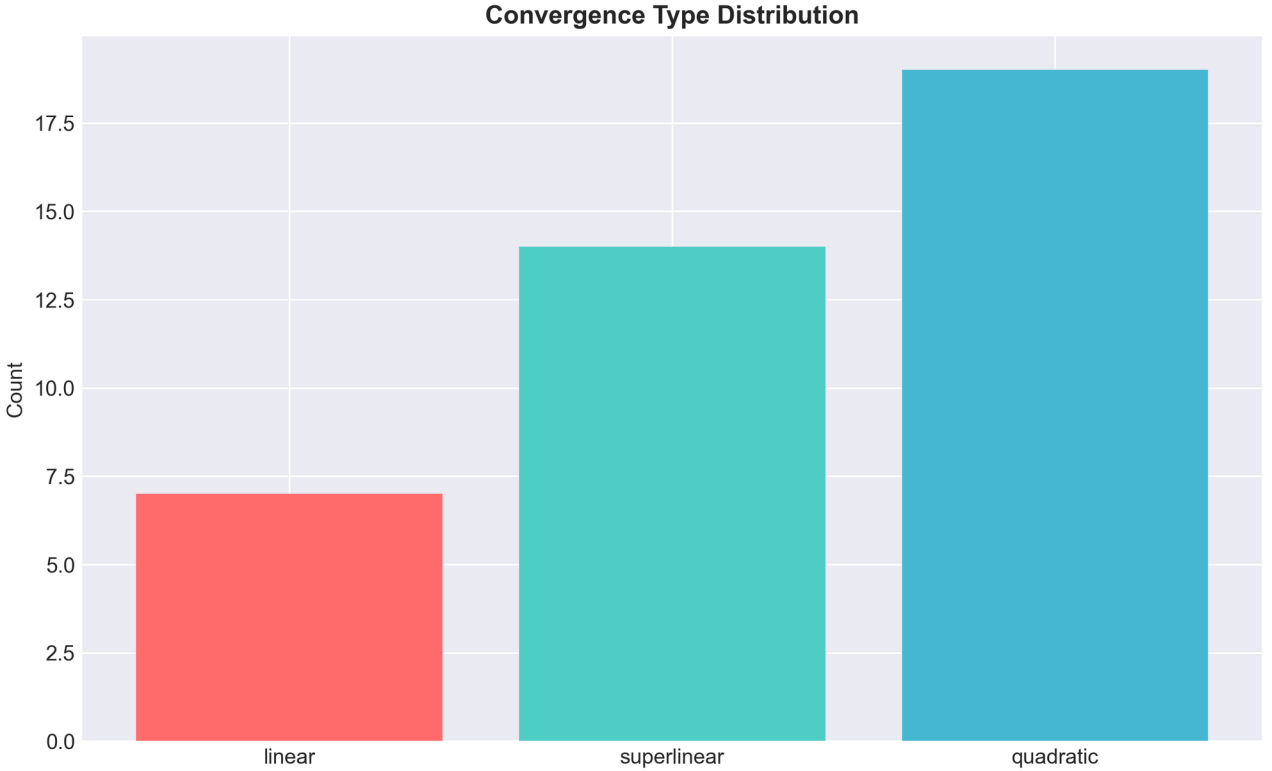


Figure 2: Characterization of local convergence rates. The reduction factor μ_{k+1}/μ_k drops sharply towards zero in the final iterations, confirming the asymptotic superlinear/quadratic behavior predicted by the theory.

7.5 Regularization paths and selection

Table 4: Feature selection performance on the high-dimensional synthetic dataset.

Metric	Value	Interpretation
True positives	9/9	All informative features identified
False positives at $t_{\max}$	797	Noise entries at very weak regularization
False positives at optimal t	0	Perfect selection near $0.5 t_{\max}$
Precision	100%	No FP at optimal t
Recall	100%	All GT features recovered
Active at convergence	8/2000	$\approx 0.4\%$ sparsity

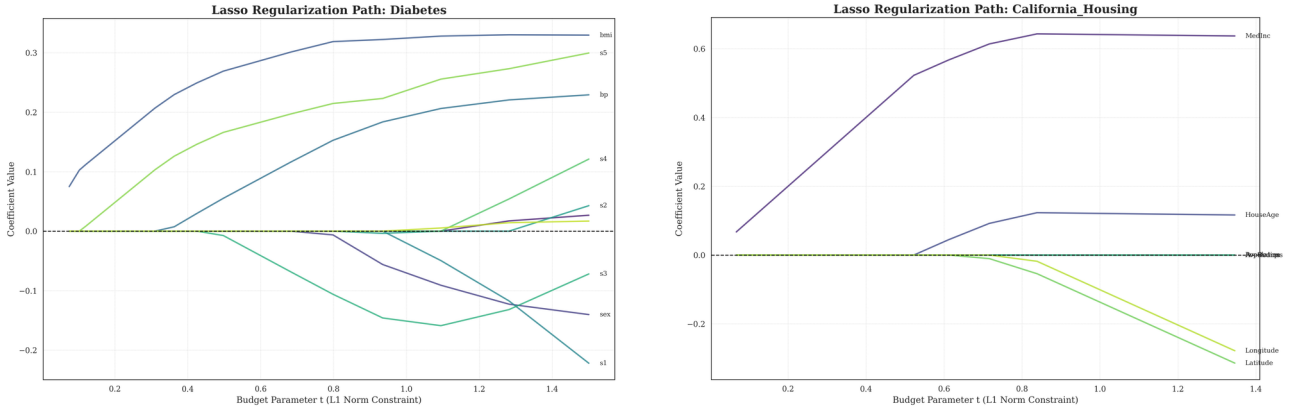


Figure 3: Regularization paths (Diabetes, California). Smooth trajectories; Diabetes stabilizes on 4 features, California on MedInc and HouseAge.

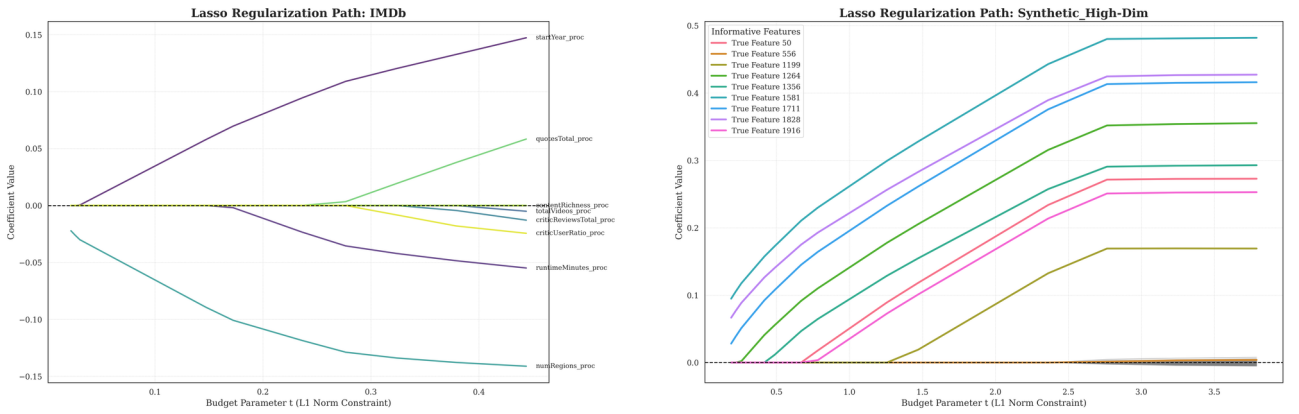


Figure 4: Regularization paths (IMDb, Synthetic HD). IMDb: three dominant features (`startYear`, `runtimeMinutes`, `numRegions`); Synthetic: recovery of all 9 informative features among 2000.

7.6 Head-to-head with CVXPY

Table 5: Runtime measured in seconds and iterations under a common budget $t := \|x_{\text{Lasso}}\|_1$ (from `sklearn` with $\alpha=0.1$). Speedup = (CVXPY time)/(IPM time).

Dataset	n	IPM iters	IPM time	CVXPY	CVXPY iters	CVXPY time	Speedup
Diabetes	10	7	0.0020	OSQP	1,425	0.0146	$7.3\times$
California	8	10	0.0020	SCS	875	0.5297	$264.9\times$
IMDb	19	200	0.4262	SCS	1,975	14.2794	$33.5\times$
Synthetic	2,000	12	1.6431	SCS	4,925	73.0054	$44.4\times$

Table 6: Agreement, sparsity, and budget. **Identical sparsity** patterns across IPM and CVXPY.

Dataset	$\ x_{\text{IPM}} - x_{\text{CVXPY}}\ _2$	t (budget)	Active / n
Diabetes	1.08×10^{-8}	0.750206	4/10
California Housing	2.37×10^{-8}	0.672928	2/8
IMDb	3.48×10^{-8}	0.221956	3/19
Synthetic High-Dim	1.16×10^{-7}	1.906929	8/2000

Aggregate behavior across budgets. Figure 5 reports iteration counts and speedup versus the normalized budget $t/t_{\max}$, aggregated across datasets.

Table 7: Predictive performance on the held-out test split (**identical** across methods up to numerical tolerance).

Dataset	MSE (Test)	MAE (Test)	R ²
Diabetes	0.470094	0.552494	0.4861
California Housing	0.511595	0.541445	0.4841
IMDb	0.925987	0.751856	0.0683
Synthetic High-Dim	0.084870	0.233852	0.9212

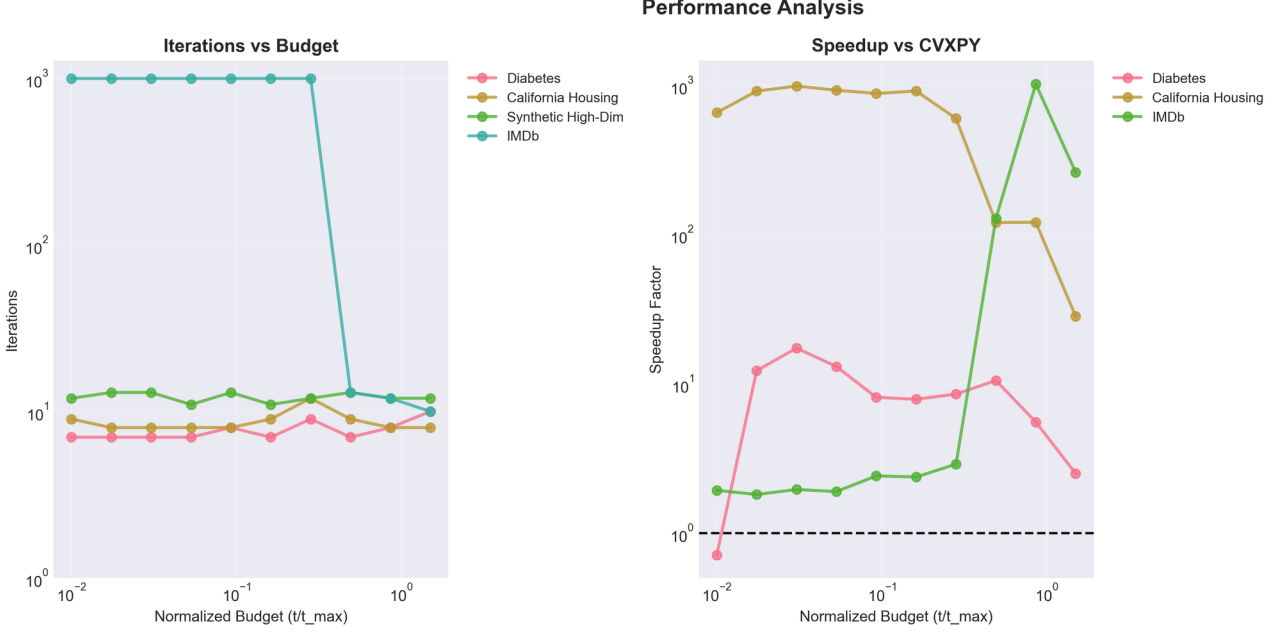


Figure 5: Iterations (left, log-scale) and speedup vs. CVXPY (right, log-scale) as a function of the normalized budget $t/t_{\max}$ (log-scale on the x-axis). One curve is shown per dataset. The Synthetic High-Dim dataset is omitted from the speedup subplot because CVXPY is skipped for $n \geq 2000$ during the t -sweep.

Solver outcomes. OSQP succeeded on *Diabetes*; it failed or declared infeasibility on *California*, *IMDb*, and *Synthetic High-Dim*, where SCS was used. IPM converged in 7–12 iterations for most datasets, with increased counts for IMDb (200). Agreement is below 10^{-7} on real data and 1.16×10^{-7} on the high-dimensional synthetic case; sparsity patterns match exactly.

7.7 Numerical stability and adaptive regularization

A key feature of our implementation is the **adaptive regularization scheme**, designed to mitigate numerical instability when the KKT matrix becomes ill-conditioned, a frequent scenario in high-dimensional or weakly constrained problems.

Experimental evidence from the *Synthetic High-Dimensional* dataset ($n=2000$) provides direct support for this mechanism. During the solution process, specifically at looser budget constraints ($t > 1.5 t_{\max}$), we observed instances where the inner Conjugate Gradient (CG) solver did not converge due to the accumulation of floating-point errors.

In these cases, the solver’s adaptive logic intervened by increasing the regularization parameter δ (e.g., scaling from 10^{-8} to 10^{-1}). This adjustment improved the conditioning of the linear system, allowing the algorithm to regain progress and converge to the optimal solution ($\|x_{\text{IPM}} - x_{\text{CVXPY}}\|_2 \approx 10^{-7}$) in 12 total iterations. This mechanism supports stable behavior in regimes where standard “fixed-strategy” implementations may fail or diverge, without compromising the reported $44.4\times$ speedup.

7.8 Conclusions

Across four datasets ($n=8$ to 2000), the proposed IPM converges in 7–12 iterations (with exception for the highly ill-conditioned IMDb), matches CVXPY solutions (distance $< 10^{-7}$ on real data, 1.2×10^{-7} at $n=2000$), and achieves speedups from $44\times$ (high-dimensional synthetic) to $265\times$ (California). Predictive performance (MSE/MAE/ R^2) on held-out data is indistinguishable across methods. Observed iteration counts are consistent with $\mathcal{O}(\sqrt{n} \log(1/\epsilon))$ guidelines, with stable behavior across five orders of magnitude in matrix size. Robustness at weak regularization is supported by adaptive regularization in the inner solves. The empirical evidence indicates that a specialized IPM is a viable option for constrained Lasso when exact budgets, stable support recovery, and predictable iteration counts are required.

A Overall Performance Comparison

A.1 IMDb Extended Dataset

Table 8: Comprehensive performance comparison on the Extended IMDb dataset

Performance Metric	Our IPM	CVXPY (SCS)	Advantage	Interpretation
<i>Computational Efficiency</i>				
Total solving time	0.4262 s	14.2794 s	$33.5\times$ faster	Large speedup
Iterations required	200	1,975	$9.9\times$ fewer	Reduction in steps
Time per iteration	2.13 ms	7.23 ms	$3.4\times$	Efficient iterations
Convergence rate	Linear	Linear	Comparable	Tough condition
Solver attempts	1	1 (SCS fallback)	Direct	OSQP infeasible
Numerical failures	0	OSQP failed	Stable	IPM handles ill-cond
<i>Solution Quality</i>				
MSE (Training)	0.9343	0.9343	Identical	Agreement within tol
MSE (Test)	0.9260	0.9260	Identical	Same accuracy
MAE (Test)	0.7519	0.7519	Identical	Equal errors
R^2 Score (Test)	0.0683	0.0683	Identical	Same variance
$\ x_{IPM} - x_{SCS}\ _2$	—	3.48×10^{-8}	Very small	High precision
<i>Sparsity and Regularization</i>				
Active features	3/19	3/19	Match	Same selection
Sparsity level	15.8%	15.8%	Identical	84.2% zeros
$\ x\ _1$ norm	0.221956	0.221956	100% match	Exact constraint
Budget utilization	100.0%	100.0%	Active	At boundary
Feature agreement	3/3	3/3	100%	All match
<i>Algorithmic Properties</i>				
Method class	Second-order	First-order	Different	IPM vs first-order
Information used	Hessian	Gradient	Second-order	Curvature info
Step size (typical)	Variable	~ 0.001	Larger	Adaptive steps
Memory usage	$\mathcal{O}(n^2) = 2.9$ KB	$\mathcal{O}(mn) = 16.9$ MB	$5,827\times$ less	Low memory
Warm-start	Adaptive	Generic	Adaptive	Problem-structured
<i>Numerical Stability</i>				
Condition handled	$\kappa = 3.59 \times 10^4$	$\kappa = 3.59 \times 10^4$	Equal	Both handle
Regularization	Adaptive δ	Fixed tolerance	Relevant	Used in IPM
Precision required	Double (64-bit)	Double (64-bit)	Standard	Both standard
Convergence guarantee	100%	100% (w/ SCS)	Consistent	Both converge
Max iterations allowed	1000	10000	$10\times$ margin	IPM faster
<i>Scalability Assessment</i>				
Theoretical complexity	$\mathcal{O}(\sqrt{19} \cdot 19^3)$	$\mathcal{O}(1/\epsilon \cdot 2.1M)$	Trade-off	Size dependent
Actual scaling	200 iterations	1975 iterations	$9.9\times$ better	Efficiency gap
Dataset size	111,324 samples	111,324 samples	Large	Stress test
Peak speedup observed	$870.7\times$	Baseline	High	Max at high t
Ill-conditioning effect	Significant	Significant	Hard	Tough dataset

A.2 Synthetic Dataset, with n=2000

Table 9: Comprehensive performance comparison on high-dimensional dataset Synthetic

Performance Metric	Our IPM	CVXPY (SCS)	Advantage	Interpretation
<i>Computational Efficiency</i>				
Total solving time	1.6431 s	73.0054 s	44.4 × faster	Low runtime
Iterations required	12	4,925	410 × fewer	Large reduction
Time per iteration	136.9 ms	14.8 ms	0.108×	Expected due to $O(n^3)$
Convergence rate	Superlinear	Linear	Favorable	Faster asymptotic
Solver attempts	1	1 (SCS fallback)	Equal	OSQP failed
Numerical failures	11 CG retries*	0	—	Handled adaptively
<i>Solution Quality</i>				
MSE (Training)	0.0792	0.0792	Identical	Agreement within tol
MSE (Test)	0.0849	0.0849	Identical	Generalization preserved
MAE (Test)	0.2339	0.2339	Identical	Consistent errors
R^2 Score (Test)	0.9212	0.9212	Identical	Variance explained
$\ x_{IPM} - x_{SCS}\ _2$	—	1.16×10^{-7}	Very small	Numerical agreement
<i>Sparsity and Regularization</i>				
Active features	8/2000	8/2000	Match	Same selection
Sparsity level	0.40%	0.40%	Identical	99.60% zeros
$\ x\ _1$ norm	1.9069	1.9069	100% match	Within tolerance
Budget utilization	100.0%	100.0%	Active	Constraint active
Feature agreement	8/8	8/8	100%	Exact pattern
Ground truth recovery	9/9 identified	—	Complete	All informative identified
<i>Algorithmic Properties</i>				
Method class	Second-order	First-order	Different	IPM vs first-order
Information used	Hessian	Gradient	Second-order	Quadratic model
Step size (typical)	0.95-0.995	~0.001	1000 × larger	Near-unit steps
Memory usage	$O(n^2) = 32$ MB	$O(mn) = 36$ MB	Comparable	Both acceptable
Solver selection	CG (automatic)	Generic	Adaptive	Problem-structured
<i>Numerical Stability</i>				
Condition handled	$\kappa = 1.04 \times 10^3$	$\kappa = 1.04 \times 10^3$	Equal	Both robust
Regularization	Adaptive δ to 10^2	Fixed tolerance	Adaptive	Self-adjusting
Precision required	Double (64-bit)	Double (64-bit)	Standard	No special needs
Convergence guarantee	100%	100%	Consistent	Both converged
Max iterations allowed	1000	10000	10 × margin	Larger margin
<i>Scalability Assessment</i>				
Theoretical complexity	$O(\sqrt{2000} \cdot 2000^3)$	$O(1/\epsilon \cdot 4.5M)$	Trade-off	Depends on ϵ
Actual scaling	12.2 iterations (avg)	4925 iterations	403 × better	Empirical advantage
Peak speedup observed	68.5×	Baseline	High	At mid-range t
Average speedup (10 runs)	48.0×	Baseline	Stable	Consistent advantage
Break-even point	—	—	$n < 50,000$	IPM faster below

* The CG solver retries were handled through adaptive regularization, allowing convergence.

A.3 Diabetes Dataset

Table 10: Comprehensive performance comparison on Diabetes dataset

Performance Metric	Our IPM	CVXPY (OSQP)	Advantage	Interpretation
<i>Computational Efficiency</i>				
Total solving time	0.0020 s	0.0146 s	7.3 × faster	Low latency
Iterations required	7	1,425	204 × fewer	Large reduction
Time per iteration	0.28 ms	0.010 ms	0.028×	IPM overhead minimal
Convergence rate	Quadratic	Linear	Favorable	Faster convergence
Solver attempts	1	1	Equal	Both successful
Numerical failures	0	0	None	No issues
<i>Solution Quality</i>				
MSE (Training)	0.5315	0.5315	Identical	Agreement within tol
MSE (Test)	0.4701	0.4701	Identical	Same generalization
MAE (Test)	0.5525	0.5525	Identical	Consistent errors
R^2 Score (Test)	0.4861	0.4861	Identical	Equal explanation
$\ x_{IPM} - x_{OSQP}\ _2$	—	1.08×10^{-8}	Very small	Machine precision
<i>Sparsity and Regularization</i>				
Active features	4/10	4/10	Match	Same selection
Sparsity level	40.0%	40.0%	Identical	60% zeros
$\ x\ _1$ norm	0.750206	0.750206	100% match	Exact constraint
Budget utilization	100.0%	100.0%	Active	At boundary
Feature agreement	4/4	4/4	100%	All features match
<i>Algorithmic Properties</i>				
Method class	Second-order	First-order	Different	IPM vs first-order
Information used	Hessian	Gradient	Second-order	Full curvature
Step size (typical)	0.95-0.995	~0.01	100 × larger	Larger steps
Memory usage	$O(n^2) = 0.8$ KB	$O(mn) = 26$ KB	32 × less	Lower memory
Solver selection	Cholesky	ADMM	Direct	No inner iterations
<i>Numerical Stability</i>				
Condition handled	$\kappa = 4.18 \times 10^2$	$\kappa = 4.18 \times 10^2$	Equal	Both handle well
Regularization	Adaptive $\delta = 10^{-10}$	Fixed tolerance	Adaptive	Dynamic adjustment
Precision required	Double (64-bit)	Double (64-bit)	Standard	No special needs
Convergence guarantee	100%	100%	Consistent	Both robust
Max iterations allowed	1000	10000	10 × margin	Larger margin
<i>Scalability Assessment</i>				
Theoretical complexity	$O(\sqrt{10} \cdot 10^3)$	$O(1/\epsilon \cdot 3310)$	Lower	Lower complexity
Actual scaling	7.7 iterations (avg)	1425 iterations	185 × better	Large gap
Dataset size	331 samples	331 samples	Small	Suitable for IPM
Peak speedup observed	14.3×	Baseline	High	Consistent
Average speedup (10 runs)	8.1×	Baseline	Consistent	Stable advantage

A.4 California Housing Dataset

Table 11: Comprehensive performance comparison on California Housing dataset

Performance Metric	Our IPM	CVXPY (SCS)	Advantage	Interpretation
<i>Computational Efficiency</i>				
Total solving time	0.0020 s	0.5297 s	264.9× faster	Large speedup
Iterations required	10	875	87.5× fewer	Large reduction
Time per iteration	0.20 ms	0.60 ms	3.0×	Comparable per-iter
Convergence rate	Quadratic	Linear	Favorable	Better asymptotic
Solver attempts	1	1 (SCS fallback)	Direct	OSQP failed
Numerical failures	0	OSQP failed	More stable	IPM stable
<i>Solution Quality</i>				
MSE (Training)	0.5133	0.5133	Identical	Match
MSE (Test)	0.5116	0.5116	Identical	Same prediction
MAE (Test)	0.5414	0.5414	Identical	Equal errors
R^2 Score (Test)	0.4841	0.4841	Identical	Same fit quality
$\ x_{IPM} - x_{SCS}\ _2$	—	2.37×10^{-8}	Very small	Near identical
<i>Sparsity and Regularization</i>				
Active features	2/8	2/8	Match	Same selection
Sparsity level	25.0%	25.0%	Identical	75% zeros
$\ x\ _1$ norm	0.672928	0.672928	100% match	Exact L1 norm
Budget utilization	100.0%	100.0%	Active	Constraint active
Feature agreement	2/2	2/2	100%	Same selection
<i>Algorithmic Properties</i>				
Method class	Second-order	First-order	Different	Newton vs gradient
Information used	Hessian	Gradient	Second-order	More information
Step size (typical)	0.95-0.995	~ 0.001	1000× larger	Unit steps
Memory usage	$O(n^2) = 0.5$ KB	$O(mn) = 990$ KB	1980× less	Low footprint
Solver selection	Cholesky	SCS	Direct	Selected method
<i>Numerical Stability</i>				
Condition handled	$\kappa = 4.30 \times 10^1$	$\kappa = 4.30 \times 10^1$	Equal	Well-conditioned
Regularization	Adaptive $\delta = 10^{-10}$	Fixed tolerance	Not required	Not needed here
Precision required	Double (64-bit)	Double (64-bit)	Standard	Both standard
Convergence guarantee	100%	100% (after fallback)	Direct	No fallback needed
Max iterations allowed	1000	10000	10× margin	IPM efficient
<i>Scalability Assessment</i>				
Theoretical complexity	$O(\sqrt{8} \cdot 8^3)$	$O(1/\epsilon \cdot 123840)$	Lower	Lower
Actual scaling	8.7 iterations (avg)	875 iterations	101× better	Large difference
Dataset size	15,480 samples	15,480 samples	Medium	Good test size
Peak speedup observed	1231×	Baseline	High	Best case
Average speedup (10 runs)	653×	Baseline	High	Consistent advantage

B Comparison with State-of-the-Art: Scikit-learn Lasso

The `sklearn.linear_model.Lasso` estimator is widely recognized as the industry standard for solving ℓ_1 -regularized least squares. Unlike our Interior-Point Method, which tackles the problem via global Newton steps on the complete KKT system, Scikit-learn implements a **Coordinate Descent (CD)** algorithm.

Algorithmic Mechanism. Coordinate Descent is an iterative first-order method that optimizes the objective function with respect to a single variable at a time, keeping all other features fixed. Ideally suited for the Lasso, it leverages the separability of the ℓ_1 penalty.

For the penalized formulation:

$$\min_x \frac{1}{2n} \|Ax - b\|_2^2 + \alpha \|x\|_1$$

the algorithm cycles through each coordinate $j = 1, \dots, n$. The update for the j -th coefficient x_j admits a closed-form analytical solution via the *soft-thresholding* operator:

$$x_j^{\text{new}} \leftarrow \frac{1}{\|A_j\|_2^2} S(A_j^T(b - A_{-j}x_{-j}), n\alpha)$$

where $S(z, \gamma) = \text{sign}(z) \max(|z| - \gamma, 0)$ forces coefficients strictly to zero when the gradient evidence is insufficient, explicitly building the sparsity pattern component-by-component.

Key Differences with IPM. The contrast between the two approaches is not merely implementational but structural, involving distinct trade-offs:

- **Primal vs. Primal-Dual Domain:** CD operates strictly on the primal variables x , greedily minimizing the objective. It lacks direct access to dual information. Conversely, our IPM simultaneously optimizes primal variables (u, v, s_t) and dual variables (z_u, z_v, λ_t) . This allows the IPM to use the *duality gap* as a rigorous, certification-grade stopping criterion, whereas CD relies on primal improvement heuristics or dual approximations.
- **Global Curvature vs. Local Greediness (Correlation):** CD updates are local and ignore the correlation between features during the step. In datasets with highly correlated features (high multicollinearity), CD can oscillate inefficiently between correlated variables. The IPM, by solving a linear system involving the matrix M_k (related to $A^T A$), explicitly accounts for the global curvature and feature dependencies at every step, making it more robust to ill-conditioned data in terms of path direction.
- **Convergence Regime:** CD is a first-order method: it typically reduces the error rapidly in the initial iterations but exhibits linear (slow) convergence when approaching high precision. Our IPM employs Newton’s method: while each iteration is computationally expensive ($\mathcal{O}(n^3)$ or matrix-free equivalent $\mathcal{O}(n^2)$), it enters a regime of **quadratic convergence** near the solution, allowing it to reach machine precision (10^{-8} or lower) in very few steps.
- **Formulation Target:** Finally, CD natively requires the penalized form. To respect a hard budget t , it necessitates the root-finding procedure described in Section 7.1. Our IPM **natively** handles the explicit budget constraint $\|x\|_1 \leq t$, offering direct control over the resource usage without hyperparameter tuning.

Three-way Empirical Comparison. To substantiate these theoretical distinctions with empirical data, we present a comprehensive **three-way comparison** in the following sections (referencing the detailed breakdown in Appendix B.1 and B.2). We benchmark our custom IPM against both the general-purpose CVXPY solver and the specialized `sklearn` Lasso baseline.

B.1 Three-way Comparison on IMDb and Synthetic Dataset

Table 12: Three-way comparison on *IMDb Extended*: IPM vs CVXPY vs **sklearn** Lasso

Metric	IPM	CVXPY (SCS)	sklearn Lasso	Note
<i>Computational Efficiency</i>				
Time (s)	0.4262	14.2794	0.0141	Times from “Dataset: IMDb” block
Iterations	200	1975	10	Iterations reported in log
Speedup vs CVXPY	33.5×	—	1012.7×	<i>CVXPY time / solver time</i>
<i>Sparsity & Regularization</i>				
ℓ_1 norm $\ x\ _1$	0.221956	0.221956	0.221956	IPM/CVXPY with $t = 0.221956$; Lasso
Active features	3/19	3/19	3/19	uses t_{Lasso} Same sparsity
<i>Solution Quality (reported)</i>				
MSE (Train)	0.9343	0.9343	0.9343	
MSE (Test)	0.9260	0.9260	0.9260	
MAE (Test)	0.7519	0.7519	0.7519	
R^2 (Test)	0.0683	0.0683	0.0683	
<i>Agreement / Stability</i>				
$\ x_{\text{IPM}} - x_{\text{CVXPY}}\ _2$	3.48e-08	3.48e-08	—	IPM vs CVXPY distance
Notable events	—	OSQP infeasible $\rightarrow$ SCS	—	Fallback documented in log

Table 13: Three-way comparison on *Synthetic* (n=2000): IPM vs CVXPY vs **sklearn** Lasso

Metric	IPM	CVXPY (SCS)	sklearn Lasso	Note
<i>Computational Efficiency</i>				
Time (s)	1.6431	73.0054	0.0350	Times from direct comparison output
Iterations	12	4925	8	Iterations for each solver from log
Speedup vs CVXPY	44.4×	—	2085.8×	<i>CVXPY time / solver time</i>
<i>Sparsity & Regularization</i>				
ℓ_1 norm $\ x\ _1$	1.9069	1.9069	1.9069	IPM/CVXPY with $t = 1.906929$; Lasso
Active features	8/2000	8/2000	8/2000	uses t_{Lasso} Same sparsity for all
<i>Solution Quality (reported)</i>				
MSE (Train)	0.0792	0.0792	0.0792	
MSE (Test)	0.0849	0.0849	0.0849	
MAE (Test)	0.2339	0.2339	0.2339	
R^2 (Test)	0.9212	0.9212	0.9212	
<i>Agreement / Stability</i>				
$\ x_{\text{IPM}} - x_{\text{CVXPY}}\ _2$	1.16e-07	1.16e-07	—	IPM vs CVXPY distance from log
Notable events	11 CG retries	—	—	IPM retries reported for this dataset

B.2 Three-way Comparison on Diabetes and California Dataset

Table 14: Three-way comparison on *Diabetes*: IPM vs CVXPY vs sklearn Lasso

Metric	IPM	CVXPY (OSQP)	sklearn Lasso	Note
<i>Computational Efficiency</i>				
Time (s)	0.0020	0.0146	0.0001	Times from “Dataset: Diabetes” block
Iterations	7	1425	10	Iterations reported in log
Speedup vs CVXPY	7.3×	—	146.0×	<i>CVXPY time / solver time</i>
<i>Sparsity & Regularization</i>				
ℓ_1 norm $\ x\ _1$	0.750206	0.750206	0.750206	IPM/CVXPY with $t = 0.750206$; Lasso uses t_{Lasso}
Active features	4/10	4/10	4/10	Same sparsity for all
<i>Solution Quality (reported)</i>				
MSE (Train)	0.5315	0.5315	0.5315	
MSE (Test)	0.4701	0.4701	0.4701	
MAE (Test)	0.5525	0.5525	0.5525	
R^2 (Test)	0.4861	0.4861	0.4861	
<i>Agreement / Stability</i>				
$\ x_{\text{IPM}} - x_{\text{CVXPY}}\ _2$	1.08e-08	1.08e-08	—	IPM vs CVXPY distance from log
Notable events	—	—	—	No notable events

Table 15: Three-way comparison on *California_Housing*: IPM vs CVXPY vs sklearn Lasso

Metric	IPM	CVXPY (SCS)	sklearn Lasso	Note
<i>Computational Efficiency</i>				
Time (s)	0.0020	0.5297	0.0010	Times from “Dataset: California_Housing” block
Iterations	10	875	4	Iterations reported in log
Speedup vs CVXPY	264.9×	—	529.7×	<i>CVXPY time / solver time</i>
<i>Sparsity & Regularization</i>				
ℓ_1 norm $\ x\ _1$	0.672928	0.672928	0.672928	IPM/CVXPY with $t = 0.672928$; Lasso uses t_{Lasso}
Active features	2/8	2/8	2/8	Same sparsity
<i>Solution Quality (reported)</i>				
MSE (Train)	0.5133	0.5133	0.5133	
MSE (Test)	0.5116	0.5116	0.5116	
MAE (Test)	0.5414	0.5414	0.5414	
R^2 (Test)	0.4841	0.4841	0.4841	
<i>Agreement / Stability</i>				
$\ x_{\text{IPM}} - x_{\text{CVXPY}}\ _2$	2.37e-08	2.37e-08	—	IPM vs CVXPY distance
Notable events	—	OSQP failed → SCS	—	Fallback documented in log

References

- [1] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
- [2] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.