2024-2025

Continuous Optimization

INFO-F524 Project : Technical report

Thomas Vray

Wassim Al Khouri

Dimitri Papadimitriou

23/05/2025

Table des matières

1 Introduction

In order to design a new solver to perform regression analysis, we studied and implemented the performance of different iterative algorithms. First, we studied the most common and used ones, ISTA and FISTA. The objective of the project is to provide convincing arguments in favor of FISTA. In addition to the theoretical proofs in favor of FISTA versus ISTA, we tested these two algorithms on several datasets with different regularization problems. We then explored the possibility of expanding our solver to other regularization problems using other algorithms (like DUAL FISTA, quasi Newton and projected methods).

2 Mathematical Background

2.1 Backtracking Line Search

2.1.1 Motivation

In many iterative methods (e.g., gradient descent or quasi-Newton), once a search direction d_k is chosen at x_k, we must select a step size $t_k > 0$ so that

$$x_{k+1} = x_k + t_k\, d_k$$

yields sufficient decrease in the objective f. A too-small t_k slows convergence, while a too-large one may violate descent or even increase f.

2.1.2 Armijo (Sufficient Decrease) Condition

One seeks t_k satisfying the Armijo rule

$$f(x_k + t_k d_k) \;\le\; f(x_k) \;+\; c\, t_k \nabla f(x_k)^\top d_k,$$

where $0 < c < 1$ is a small constant (often $c \approx 10^{-4}$). This ensures the actual decrease in f is at least a fraction c of the linearized decrease.

2.1.3 Backtracking Algorithm

A simple and robust strategy is to start from an initial t (e.g. $t = 1$) and iteratively shrink it by a factor $0 < \beta < 1$ (e.g. $\beta = 0.5$) until the Armijo condition holds.

Algorithm 1 Backtracking Line Search

```
Require: Current point $x_k$, descent direction $d_k$, parameters $c \in (0,1)$, $\beta \in (0,1)$, initial $t \leftarrow 1$.
Ensure: Step length $t_k$.
1: while $f(x_k + t\,d_k) > f(x_k) + c\,t\nabla f(x_k)^\top d_k$ do
2:     $t \leftarrow \beta\,t$
3: end while
4: $t_k \leftarrow t$
```

2.1.4 Choice of Parameters

— c controls how much decrease is required ; smaller $c \to$ easier acceptance.

— β controls the shrink rate ; common choices are $\beta = 0.5$ or 0.8.

— Initial t may come from a heuristic (e.g. $t = 1$) or a previous iteration.

2.1.5 Remarks

1. Guarantees a sufficient decrease without evaluating higher-order derivatives.
2. Combined with any descent direction d_k, it preserves global convergence under mild conditions.
3. Cost per line search is one objective and one directional-derivative evaluation per trial t.

2.2 ISTA

ISTA (Iterative Shrinkage-Thresholding Algorithm) and FISTA (Fast ISTA) are optimization algorithms designed to solve composite problems of the form :

$$\min_{x\in\mathbb{R}^n} F(x) := f(x) + g(x)$$

where :

— $f : \mathbb{R}^n \to \mathbb{R}$ is convex, differentiable, and has a Lipschitz continuous gradient with constant $L > 0$,

— $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is convex, but possibly non-smooth (e.g., ℓ_1-norm).

A common application is the LASSO problem :

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

2.3 Gradient Descent for Smooth Term

The smooth part $f(x)$ can be minimized using gradient descent :

$$x^{k+1} = x^k - \alpha\nabla f(x^k)$$

where $\alpha \in (0, \frac{1}{L}]$ is the step size. However, this method cannot directly handle the non-smooth part $g(x)$.

2.4 Proximal Operator

To incorporate the non-smooth component g, we use the **proximal operator** :

$$\text{prox}_{\alpha g}(v) := \arg\min_{x} \left\{ \frac{1}{2\alpha} \|x - v\|^2 + g(x) \right\}$$

In the case where $g(x) = \lambda \|x\|_1$, the proximal operator becomes the **soft-thresholding** operator :

$$[\text{prox}_{\alpha\lambda\|\cdot\|_1}(v)]_i = \text{sign}(v_i) \cdot \max(|v_i| - \alpha\lambda, 0)$$

2.5 FISTA

FISTA stands for fast iterative Shrinkage Algorithm which is non other than an upgrade of ISTA. It solves the same minimization problems but the convergence is proven to be $O(1/k^2)$. The idea of FISTA is to use a momentum term to speed up the convergence.

Algorithm 2 FISTA

```
1: **Input :** Initial point $x^0 = y^1 \in \mathbb{R}^n$, step size $t \le 1/L$, where $L$ is the Lipschitz constant of $\nabla f$
2: Set $k = 1$, $t_1 = 1$
3: **while** not converged **do**
4:     $x^k = \text{prox}_{tg}\left(y^k - t\nabla f(y^k)\right)$
5:     $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
6:     $y^{k+1} = x^k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x^k - x^{k-1})$
7:     $k \leftarrow k + 1$
8: **end while**
```

The proximal operator used in Step 4 is defined as :

$$\text{prox}_{tg}(v) := \arg\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2t} \|x - v\|^2 + g(x) \right\}. \tag{1}$$

FISTA achieves an improved convergence rate over ISTA. Specifically, if $F(x)$ is convex and f has a Lipschitz-continuous gradient with constant L, then FISTA guarantees :

$$F(x^k) - F(x^*) \le \frac{2L\|x^0 - x^*\|^2}{(k+1)^2}, \tag{2}$$

where x^* is an optimal solution. This is a significant improvement over ISTA, which only guarantees an $\mathcal{O}(1/k)$ rate.

2.6 Newton's Method

Newton's method is a classical optimization technique for finding a local minimum of a twice-differentiable function $f:\mathbb{R}^n \to \mathbb{R}$. Starting from x_0, it iterates

$$x_{k+1} = x_k - H_k \nabla f(x_k), \quad H_k = \left[\nabla^2 f(x_k)\right]^{-1},$$

where $\nabla f(x_k)$ is the gradient and $\nabla^2 f(x_k)$ the Hessian. Although Newton's method converges quadratically near the solution, forming and inverting $\nabla^2 f(x_k)$ costs $O(n^3)$, which scales poorly for large n.

2.7 Quasi-Newton Methods

Quasi–Newton methods avoid Hessian inversion by building an approximation $B_k \approx H_k$. The update step is

$$x_{k+1} = x_k - \alpha_k B_k \nabla f(x_k),$$

with step length α_k from a line search, and B_{k+1} chosen so that the *secant equation*

$$B_{k+1} y_k = s_k, \qquad s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k),$$

holds. This ensures first-order consistency with the true Hessian.

2.8 BFGS Algorithm

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) update maintains symmetry and positive definiteness of B_k. Define

$$\rho_k = \frac{1}{y_k^\top s_k}.$$

Then

$$B_{k+1} = \left(I - \rho_k s_k y_k^\top\right) B_k \left(I - \rho_k y_k s_k^\top\right) + \rho_k s_k s_k^\top,$$

and one sets $H_{k+1} = B_{k+1}$.

2.9 Limited-Memory BFGS (L-BFGS)

In large-scale settings storing $B_k \in \mathbb{R}^{n\times n}$ is infeasible. L-BFGS retains only the most recent m pairs $\{(s_i, y_i)\}_{i=k-m}^{k-1}$ and their scalars $\rho_i = 1/(y_i^\top s_i)$. The search direction $d_k \approx -B_k \nabla f(x_k)$ is computed by the two-loop recursion :

Algorithm 3 Two-Loop Recursion for $d_k = -H_k \nabla f(x_k)$

```
Require: Gradient $\nabla f(x_k)$, stored pairs $\{(s_i, y_i, \rho_i)\}_{i=k-m}^{k-1}$
Ensure: Search direction $d_k$
 1: $q \leftarrow \nabla f(x_k)$
 2: for $i = k-1, \dots, k-m$ do
 3:     $t \leftarrow \rho_i \, s_i^\top q$
 4:     $q \leftarrow q - t \, y_i$
 5: end for
 6: $r \leftarrow B_k^0 q$
 7: for $i = k-m, \dots, k-1$ do
 8:     $b_i \leftarrow \rho_i \, y_i^\top r$
 9:     $r \leftarrow r + s_i \, (t - b_i)$
10: end for
11: $d_k \leftarrow r$
```

Algorithm 4 L-BFGS Method

```
Require: $x_0$, memory $m$, tolerance $\varepsilon$
 1: for $k = 0, 1, 2, \dots$ do
 2:     if $\|\nabla f(x_k)\| \le \varepsilon$ or other convergence criteria met then
 3:         break
 4:     end if
 5:     compute $d_k$ via two-loop recursion
 6:     choose $t_k > 0$ by line search
 7:     $x_{k+1} \leftarrow x_k + t_k \, d_k$
 8:     $s_k \leftarrow x_{k+1} - x_k, \quad y_k \leftarrow \nabla f(x_{k+1}) - \nabla f(x_k)$
 9:     $\rho_k \leftarrow 1/(y_k^\top s_k)$
10:     if $\#\{(s_i, y_i)\} > m$ then
11:         drop oldest pair
12:     end if
13:     store $(s_k, y_k, \rho_k)$
14: end for
```

3 Regularized regression models

ISTA and FISTA can resolve different minimization problems. Here are the main ones ;

3.1 LASSO (l1 regularization)

The LASSO (Least Absolute Shrinkage and Selection Operator) is a regularized regression method designed to perform both parameter estimation and variable selection in linear models. It minimizes at the same time the least squares but it also introduces the l1 norm penalty. LASSO produces sparse solutions (solutions with many zero coefficients).

$$\frac{1}{2}||Y - X\beta||_2^2 + \lambda||\beta||_1 \tag{3}$$

ISTA can be used to compute LASSO since f (least squares part) is convex, differentiable and has a continuous gradient with constant L > 0 and g is convex.

3.2 Ridge regression (l2 regularization)

Just like LASSO, ridge regression adds a penalty term to the least squares. This penalty term is proportional to the square of the coefficient magnitude.

Unlike LASSO, l2 does not lead to sparse solutions. It does not eliminate the coefficients. Ridge regression is useful to prevent overfitting and improve robustness of the model, especially for multicollinearity.

3.3 Elastic net (l1-l2 regularization)

Elastic net is a mixture of l1 and l2 regularization. The algorithm adds the two penalty terms from the LASSO and the ridge regression. While LASSO will select only one of the n highly correlated variables, the elastic net will retain multiple owing to the squared penalty term.

4 Solver design

We have designed our solver iteratively. First, we tried to solve the ELASTIC-NET regression problem using ISTA and FISTA. We could solve any problem of the form :

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda_1\|x\|_1 + \lambda_2\|x\|^2 \tag{4}$$

We implemented them with backtracking and with constant step size t_k. We coded the shrink function which is the proximal of the l1 term. Then, we added the LBFGS function to minimize smooth functions. As for ISTA and FISTA we used backtracking and constant step size. We design our solver such that the arguments for L_BFGS are the smooth function to minimize and its gradient.

4.1 The new algorithm

As discussed in the project description, we implemented a variant of FISTA in which the sub-problem (the computation of the proximal operator) is solved with a quasi-Newton method, namely L-BFGS. The motivation is to apply the proximal operator to complex functions whose closed-form prox is unknown or too costly to store. Unfortunately, L-BFGS only applies to smooth functions, whereas FISTA is designed for composite problems in which the second term can be non-differentiable but has an easy proximal mapping. Consequently, the proposed algorithm is only useful when both terms are smooth—in which case one might as well run plain gradient descent or, better, L-BFGS on the whole objective, which (as we will show in the results section) yields superior performance.

4.2 Algorithm Input/Output Specification

ISTA

Input : `double *x` – Initial solution (vector of 0)

`Problem problem` – Struct containing data matrix A, vector b, and parameters

Output : `double *x` – Solution vector

FISTA

Input : Same as ISTA

Output : `double *x` – Solution vector (overwritten in-place)

L-BFGS

Input : `double *x` – Initial guess

`int m` – Memory parameter for L-BFGS

`Problem problem` – Struct containing data and function pointers

Output : `double *x` – Solution vector

LBFGS-FISTA

Input : `double *x` – Initial guess

`int m` – L-BFGS memory

`Problem problem` – Struct with data, parameter and optimization functions and their gradient

Output : `double *x` – Solution vector

4.3 Parameters

We have a header file "parameter.h" where most of the parameters can be changed. It centralizes the configuration for regularization, algorithmic tolerances, and line search, making it easy to adjust solver behavior without modifying the main code.

— λ_1 : l1 regression term

— λ_2 l2 regression term

— c : Armijo condition

— α : the contraction factor

— t_0 : initial step size

— ϵ : the tolerance

5 Data Selection

To evaluate our optimisation algorithms under a variety of conditions we curated a suite of *four* regression data sets : one synthetic problem and three publicly available real-world benchmarks. Table 1 summarises their key statistics (#samples n and #features p).

Boston Housing Classic small-scale benchmark ($n = 500$, $p = 14$) relating socio-economic and environmental indicators to median house prices.[fed21]

Red Wine Quality Medium-size data set ($n = 1600$, $p = 11$) in which physicochemical measurements of Portuguese red wines are mapped to sensory quality scores.[Lea18]

Gas Turbine Emissions Industrial data set ($n = 7000$, $p = 10$) recording operating conditions of a gas turbine together with the resulting NO_x emissions.[Rep19]

TABLE 1: Overview of the evaluation data sets.

Data set	Samples (n)	Features p
Synthetic	100	100
Boston Housing	500	14
Red Wine Quality	1600	11
Gas Turbine Emissions	7000	10

Rationale. This selection spans four orders of magnitude in sample size and a two-fold range in dimensionality, while covering synthetic, socio-economic, chemical, and industrial domains. Such diversity allows us to stress-test the algorithms under both noise-free and noisy conditions, with varying degrees of multicollinearity and feature scaling, thereby providing a comprehensive assessment of their robustness and practical usefulness. We used the synthetic dataset mainly for the development phase to test on simple and various cases.

6 Results and Analysis

We benchmarked the algorithms introduced above on the datasets described in Section 5. Three regularised regression problems were considered—*ridge*, *LASSO*, and *Elastic Net*—and performance was evaluated by the number of outer iterations required to reach convergence as well as the wall–clock time per iteration.

For conciseness, the main text reports the outcomes obtained on the BOSTON HOUSING dataset ($n = 500$, $p = 14$) ; complete plots for the remaining datasets appear in Appendix 8.

6.1 Line–search strategy

Our initial experiments employed a classical *backtracking line–search* to determine the step size at every outer iteration. Although this adaptive procedure enjoys solid theoretical guarantees—it satisfies the sufficient-decrease condition without requiring a priori knowledge of the Lipschitz constant L—it proved inefficient in practice : the extra objective and gradient evaluations outweighed the benefit of occasionally larger steps, resulting in longer run-times and, in several cases, a slower decrease of the objective.

Consequently, all subsequent results were obtained with a **fixed** step size equal to $1/L$, where L is a conservative upper bound on the Lipschitz constant.

6.2 Experiment 1 : Ridge regression

Ridge regression is the only setting that can be tackled by *all* four algorithms under consideration—ISTA, FISTA, L-BFGS, and the hybrid FISTA-L-BFGS. The regularisation parameter was set to $\lambda_2 = 0.1$.

Figures 1-2 show that **L-BFGS converges in far fewer iterations** than any other method. FISTA is a distant second, whereas ISTA is consistently the slowest. The hybrid FISTA-L-BFGS variant fails to make progress, indicating an implementation or modelling issue that could not be resolved within the project's time-frame. Given the similar per-iteration cost across methods (Figure 2), the iteration count is a reliable proxy for total run-time, confirming L-BFGS as the algorithm of choice for smooth objectives such as ridge regression.

6.3 Experiment 2 : LASSO

We next considered the LASSO problem, whose ℓ_1 penalty is *non-smooth*. As a result, only ISTA and FISTA apply without modification.

Consistent with theory, FISTA converges markedly faster than ISTA. The empirical convergence rates

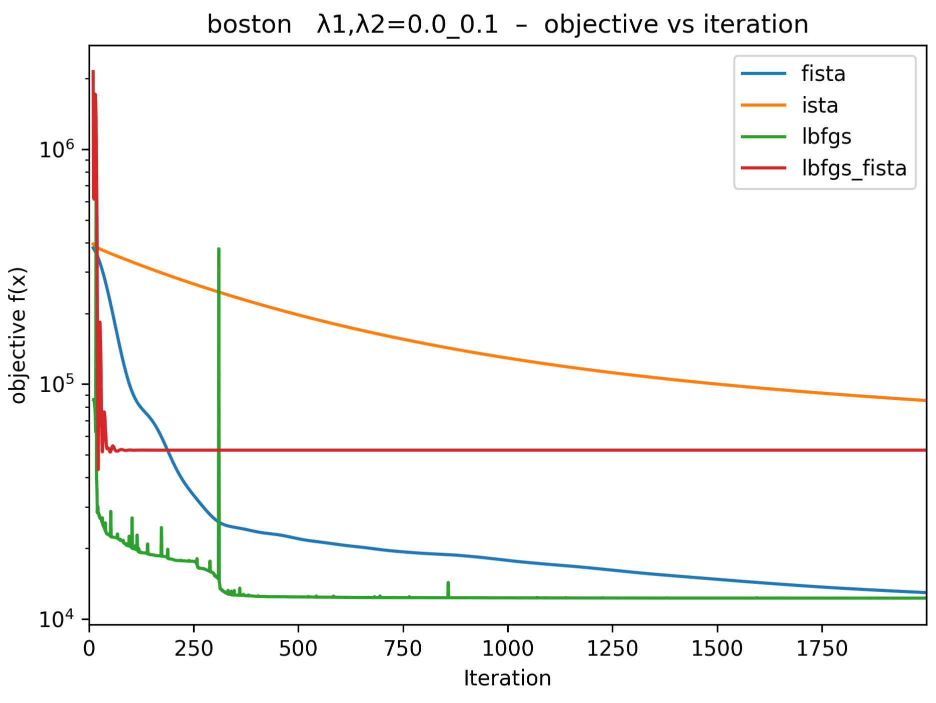


FIGURE 1: Objective value versus iterations for ridge regression on the BOSTON HOUSING dataset.

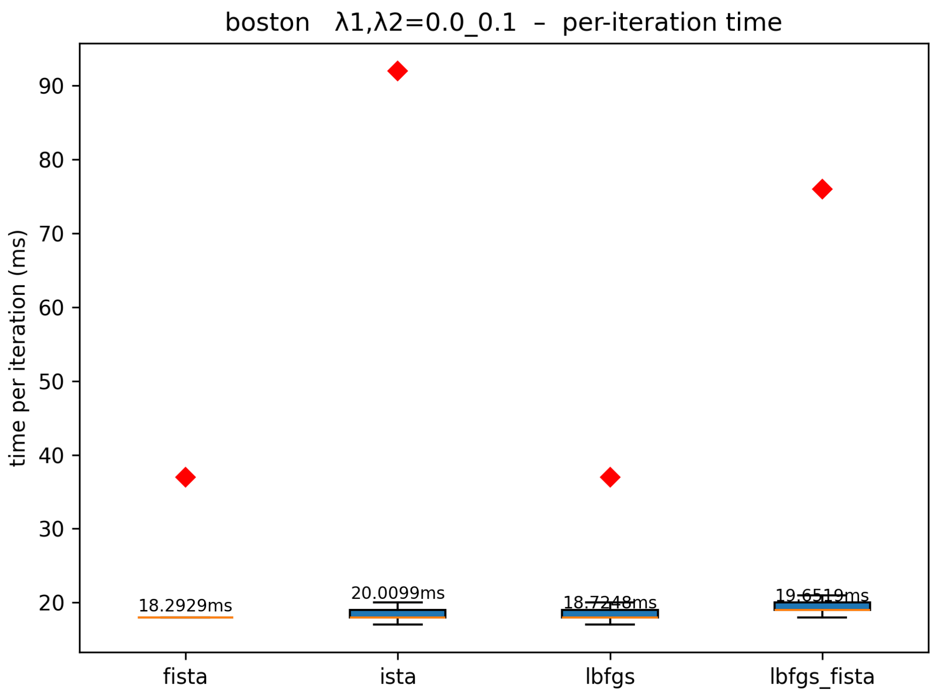


FIGURE 2: Distribution of wall–clock time per iteration for ridge regression.

(Figure 3) match the expected $\mathcal{O}(1/k^2)$ rate for FISTA and $\mathcal{O}(1/k)$ rate for ISTA, thereby validating our implementation.

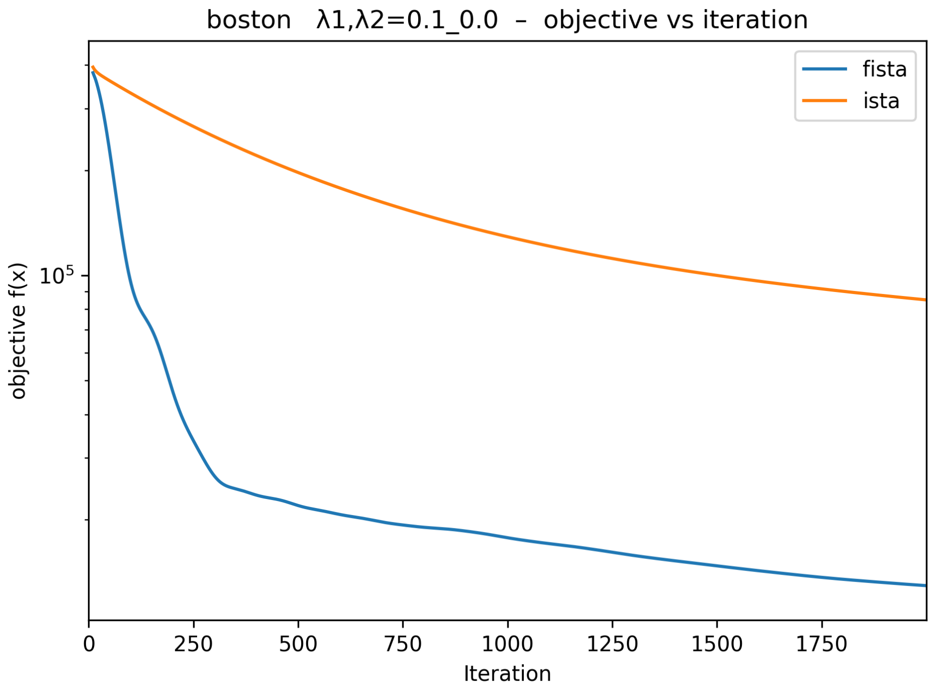


FIGURE 3: Objective value versus iterations for LASSO ($\lambda_1 = 0.1$).

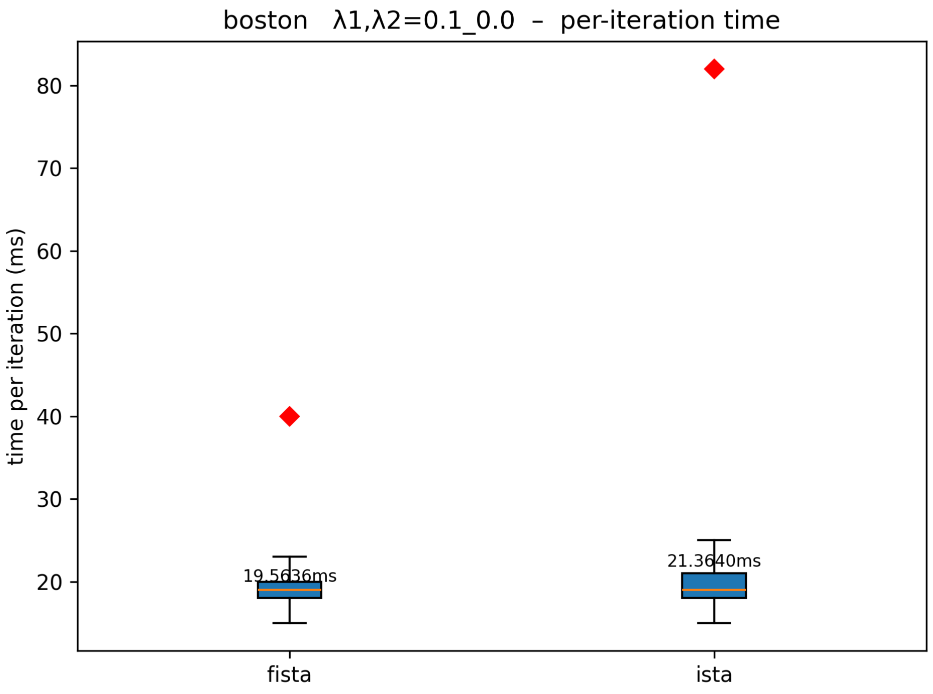


FIGURE 4: Distribution of wall–clock time per iteration for LASSO.

6.4 Experiment 3 : Elastic Net

Elastic Net combines ridge and LASSO penalties and thus inherits the non-smoothness of the ℓ_1 term; again, only ISTA and FISTA are appropriate. We set $\lambda_1 = \lambda_2 = 0.1$.

The results mirror those of the LASSO experiment : FISTA consistently outperforms ISTA in both iteration count and run-time while producing comparable objective values and sparsity patterns.

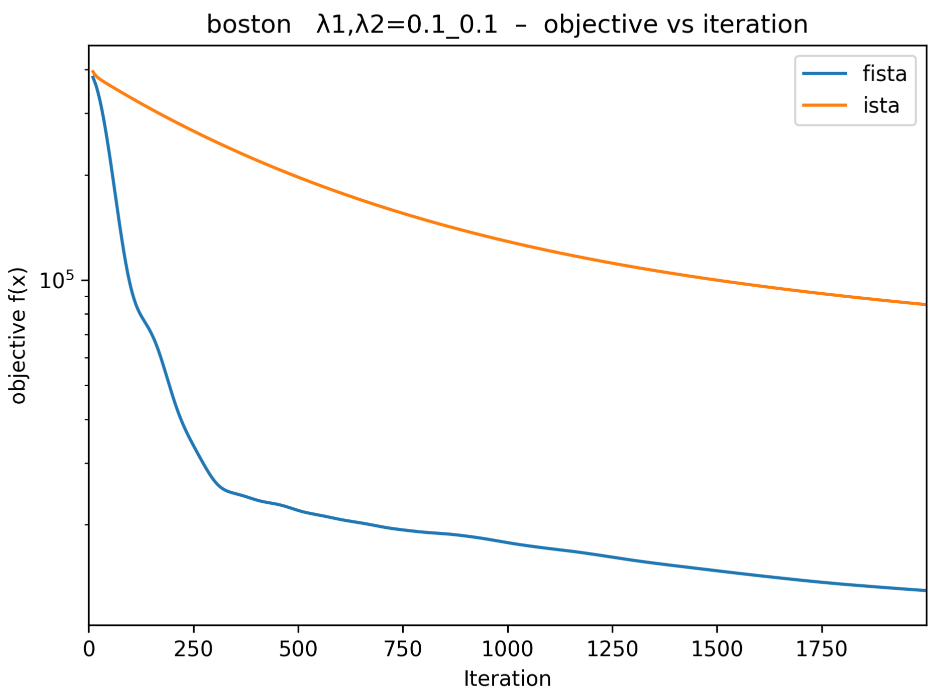


FIGURE 5: Objective value versus iterations for Elastic Net.

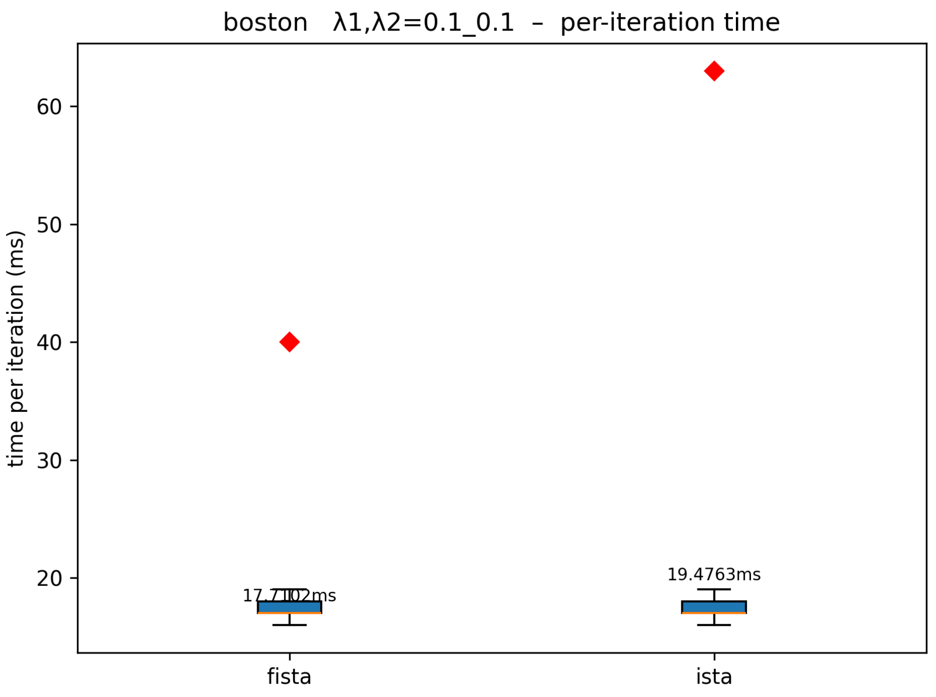


FIGURE 6: Distribution of wall–clock time per iteration for Elastic Net.

6.5 Overall findings

Our experiments lead to three main conclusions :

1. **L-BFGS is the method of choice for smooth objectives.** It reaches the optimum in the fewest iterations and, given the similar per-iteration cost across methods, the shortest wall–clock time.

2. **FISTA provides a robust and fast alternative when the objective is composite.** Its $\mathcal{O}(1/k^2)$ convergence rate offers a tangible speed-up over ISTA in problems with non-smooth penalties, such as LASSO and Elastic Net.
3. **The hybrid FISTA-L-BFGS algorithm requires further debugging.** In its current form it fails to converge even on the smooth ridge problem, suggesting an implementation error or an inappropriate parameter setting.

We did study the time for each part of FISTA and we found that most of the time spent in the function was to compute the gradient which is by far the most extensive part. The computation of the proximal was negligible as the update of the momentum term. The time for the backtracking strongly depends on the parameters chosen. Therefore, the duration of the backtracking is highly variable. It can take more time than the gradient compute for some cases while it can be negligible for other cases.

Finally, we note that the *duration of a single iteration* is remarkably stable across all methods and problem classes, reinforcing our use of the iteration count as a meaningful performance metric.

7 Conclusion

In this report, we investigated and compared several first-order and quasi-Newton optimization methods for solving regularized regression problems, specifically Ridge, LASSO, and Elastic Net. Our primary goal was to evaluate the practical performance of ISTA, FISTA, and L-BFGS under different problem structures, and to understand their convergence behavior and computational efficiency across a variety of datasets.

Our experiments confirmed the theoretical expectations :

— **L-BFGS** is the most effective solver for smooth problems such as Ridge regression, offering rapid convergence in very few iterations while maintaining low per-iteration cost.

— **FISTA** significantly outperforms ISTA for composite problems involving non-smooth terms, such as LASSO and Elastic Net. Its accelerated convergence rate provides a tangible performance advantage while retaining ease of implementation.

— **ISTA**, although conceptually simple, converges slowly in practice and is outpaced by both FISTA and L-BFGS in all tested scenarios.

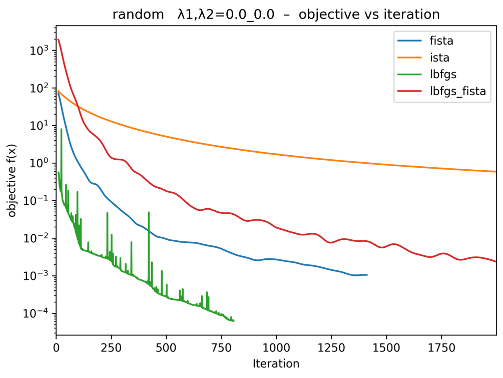
While the hybrid FISTA-LBFGS algorithm was proposed as an innovative direction for solving composite problems with complex proximals, our current implementation did not yield satisfactory results and requires further debugging and validation.

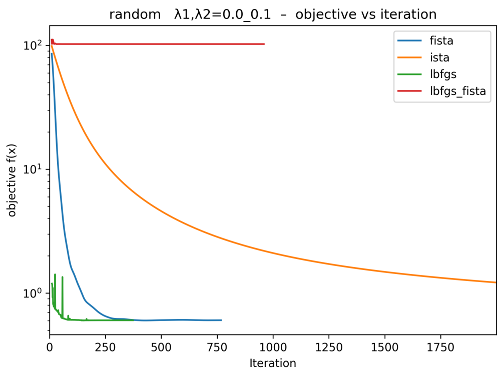
Overall, our solver demonstrates strong performance on both smooth and non-smooth regularized regression tasks. The modular design also allows for future extensions, such as support for constraint projection methods or dual formulations. Moving forward, we recommend improving the hybrid algorithm's stability and extending the benchmarking to include additional real-world datasets and larger-scale problems.

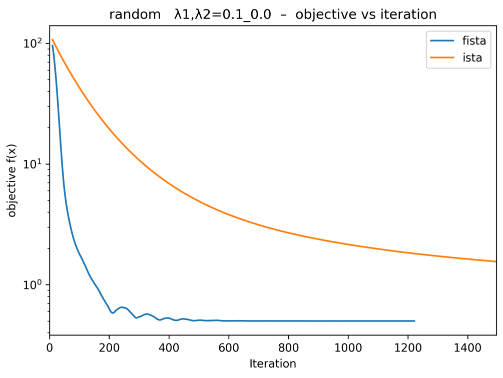
8 Annex

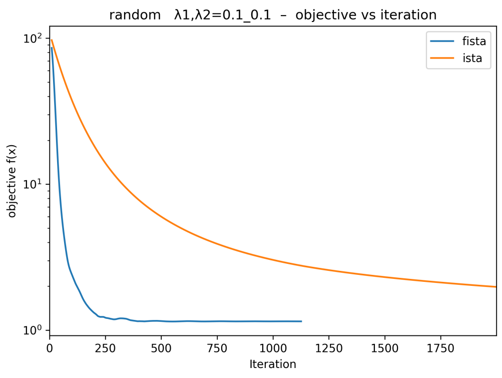
The code of our solver and all of the graphs are in the attached zip folder. We did not add the code to the report because of its size.

8.1 Randomly generated matrix Results

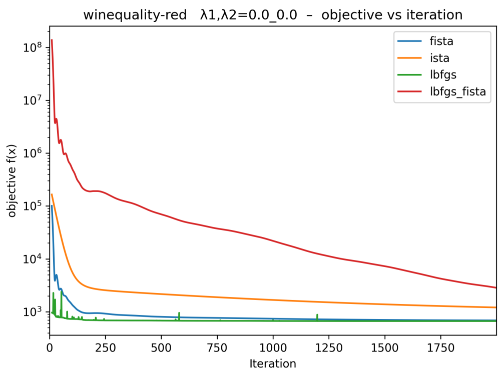


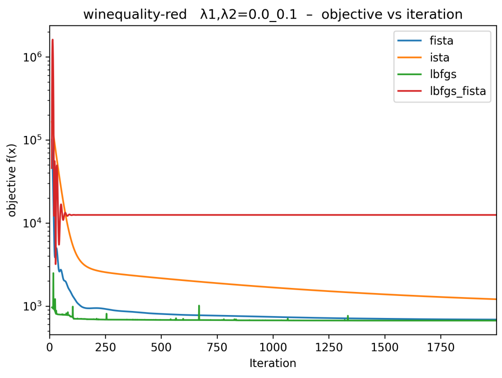


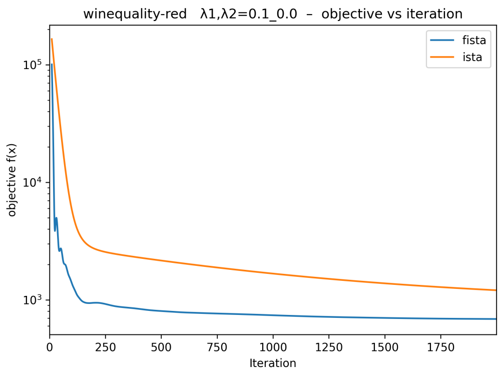


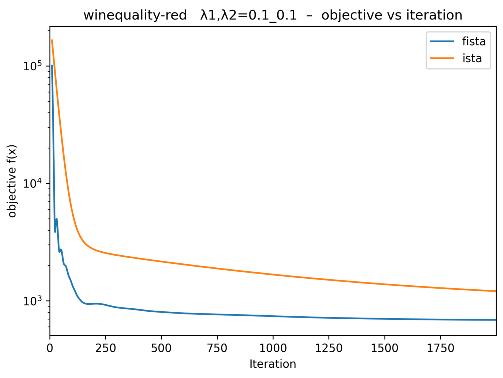


8.2 Wine Results

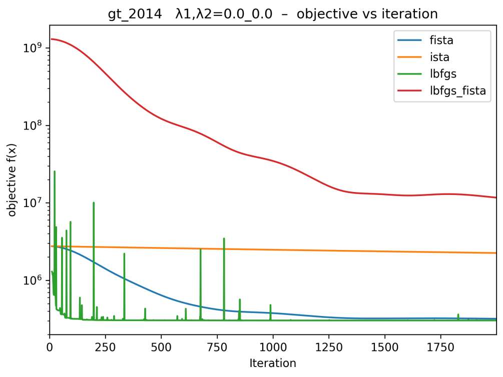


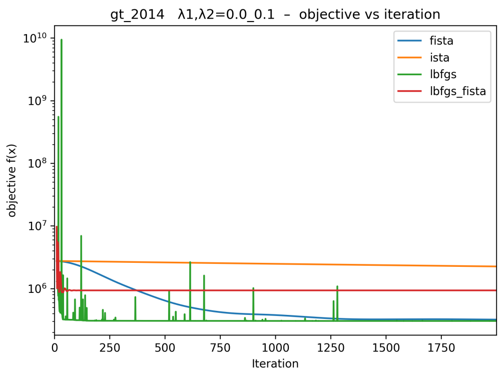


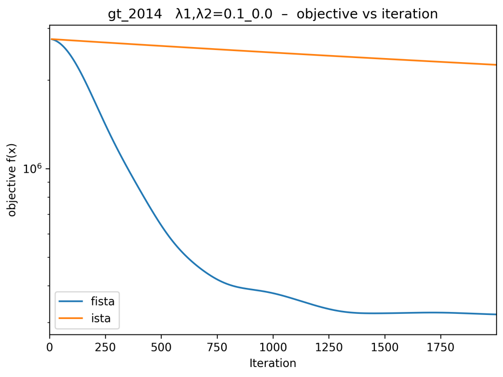


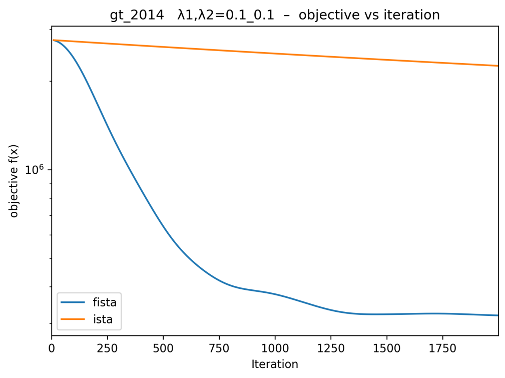


8.3 gt_2014 Results









Articles

[fed21] FEDESORIANO (2021). « Boston House Prices-Advanced Regression Techniques ». In : URL : `https://www.kaggle.com/datasets/fedesoriano/the-boston-houseprice-data`.

[Lea18] LEARNING, UCI Machine (2018). « Red Wine Quality ». In : URL : `https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009`.

[Rep19] REPOSITORY, UC Irvine Machine Learning (2019). « Gas Turbine CO and NOx Emission Data Set ». In : DOI : https://doi.org/10.24432/C5WC95.