Algorithm Design of Quasi-Newton Methods for Large-Scale Unconstrained Optimization[*]

Youthy WANG [†]

April 20, 2023

Contents

Abstract

With the rise of dimensionality (especially in deep learning, computer graphics, etc.) in recent years, people have shown concerns about large-scale problems in optimization fields. We designed a new limited-memory quasi-Newton algorithm to deal with large-scale unconstrained optimization problems under the scheme of least-square updating methods. Numerical experiments show that it is often competitive with L-BFGS and can be applied as an alternative.

[*]References of the proposal are under the standard of SIAM. For more details, please refer to Style Manual - SIAM.

[†]School of Science and Engineering, the Chinese University of Hong Kong, Shenzhen, P.R.C. Email: yuzhewang@link.cuhk.edu.cn.

1 Introduction

High dimensionality has become a main obstacle in many scientific and engineering disciplines like material science, chemistry, biology, neuroscience, geoscience, and finance in recent years. Particularly, with the rise of some novel fields in the industry, like deep learning and computer graphics, large-scale optimizations have become increasingly essential since they are the basis for many real-industry problems. Efficient enough algorithms matching the capacity of current computers serve as the key to solving large-scale unconstrained optimizations, which usually have thousands or millions of variables, with the storage and computational costs being controlled at a tolerant level. Starting from the 1980s and 90s, researchers devoted themselves to developing algorithms to solve large-scale unconstrained optimization problems, mainly by modifying known algorithms of moderate-scale ones. Some of them, such as non-linear conjugate gradient methods and limited memory quasi-Newton methods, achieved extraordinary success for some specific problem types [1], both in theory and practice. The limited memory BFGS (L-BFGS) algorithm proposed in 1989 [4], as a very competitive member of limited memory quasi-Newton methods, has been widely applied to many disciplines, for instance, physics-based animation in computer graphics [5]. Nonetheless, although L-BFGS works well in many disciplines, some other efficient algorithms may also be competitive with or even outperform it for different problem types since the scheme of quasi-Newton contains many candidates.

Motivated by this, the project aims to explore some new algorithms, and expects to

(1) design algorithms competitive with or even outperform L-BFGS for some problem types, based on the quasi-Newton updating scheme;

(2) test the algorithms with different cases to check their practical efficiency;

2 Problem Statment and Formulation

The project concentrates on developing new limited-memory algorithms under a quasi-Newton updating scheme for the following problem:

$$\min_{x\in\mathbb{R}^n} f(x),$$

where x has a large dimension (i.e., very large n) and f is in $C^1(\mathbb{R}^n)$.

Quasi-Newton methods are renowned as a collection of second-order methods for moderate-scale non-linear optimizations. Under the scheme of quasi-Newton, a sequence of approximated solutions $\{x_k\}_{k=0}^{\infty}$ is obtained by the updates at every step, that is,

$$\begin{cases} x_{k+1} = x_k - \alpha_k B_k^{-1}\nabla f(x_k) \\ B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k) \end{cases} \tag{1}$$

where the sequence $\{B_k\}_{k=0}^{\infty}$ is a collection of a symmetric estimated matrix of Hessian at every x_k [2].

Versatile quasi-Netwon algorithms appear with capricious approaches of updating B_k, among which the most famous ones are (Let $s = x_+ - x$, $y = \nabla f(x_+) - \nabla f(x)$ for simplicity.)

(i) Symmetric Rank One Correction (SR1-Correction), which corrects B_+ by adding a rank-one matrix at every update

$$B_+^{\text{SR1}} = B + \frac{(y-Bs)(y-Bs)^T}{s^T(y-Bs)}. \tag{2}$$

(ii) BFGS (one of rank two corrections), which corrects B_+ by adding a special rank-two matrix at every update

$$B_+^{\text{BFGS}} = B - \frac{Bss^TB}{s^TBs} + \frac{yy^T}{y^Ts} \tag{3}$$

(iii) DFP (one of rank two corrections), which corrects $H_+ = (B_+)^{-1}$ by adding a special rank-two matrix at every update

$$H_+ = H - \frac{Hyy^TH}{y^THy} + \frac{ss^T}{y^Ts}.$$

To keep consistent with (1), the direct update formual is

$$B_+^{\text{DFP}} = B + \frac{y^Ts + s^TBs}{(y^Ts)^2}yy^T - \frac{(Bsy^T + ys^TB)}{y^Ts} \tag{4}$$

Researchers collect some rank two corrections using convex combination BFGS and DFP updating formula to form a Broyden class, namely,

$$B_+(\theta) = (1-\theta)B_+^{\text{BFGS}} + \theta B_+^{\text{DFP}},\ 0 \le \theta \le 1, \tag{5}$$

where B_+^{BFGS} and B_+^{DFP} come from (3) and (4).

Zhang and Tewarson [2] proposed one reasonable approach that was all the rage in the 1980s by saying that B_k could be chosen by least change updates of Cholesky (LCC) factors in Frobenius norm with properly chosen weighted matrices

$$\min_{L_+} \|L_+ - L\|^2_{F,M,N},$$

where $L_+L_+^T = B_+$ and $LL^T = B$. Papers [2], and [3] gave an exact unique solution with explicit formulas to the L_+ for solving moderate-size problems. They also said that methods like BFGS and DFP were exceptional cases of the least change update scheme (i.e., rank 1 LCC), which determines another quasi-Newton updating formula mainly in theory in practice for some particular choice of M. For instance, they found one updating method in the pre-convex Broyden class (named LCCB), with flexible sizes $\{\theta_k\}$, by computing using $M = LL^T$ [2], whose efficiency was shown better than BFGS in numerical experiments [3].

Designed after the above moderate-size quasi-Newton algorithms, the most renowned limited-memory quasi-Newton method is L-BFGS, a variant BFGS update method, from its name. L-BFGS has been widely applied due to its characteristics of easy implementation, concise updating formulas, fast speed, and accurate enough solutions. One possible exposition for these advantages of L-BFGS is that its original method, BFGS, has good enough efficiency in theory and practice. Hence, it is reasonable to believe that there are some variant limited memory methods under the least update scheme in [2] enjoying an efficient performance in large-scale optimizations competitive with L-BFGS in different problem types. Our goal is to explore such algorithms.

3 Main Results

One possible limited-memory approach, is to estimate B_k in the form

$$\tau_k I_n + U_kU_k^T,\ \text{where } \tau_k \in \mathbb{R}^+, U_k \in \mathbb{R}^{n\times m}. \tag{6}$$

In every step, U_k should be updated with least change method in [2]. Namely,

$$U_{k+1} = \arg\min_{U_+} \{\|U_+ - U_k\|_{F,M}^2 : (\tau_{k+1} I_n + U_+ U_+^T) s_k = y_k\}. \tag{7}$$

Following the derivations of [2], U_+ has explicit form at every step with symmetrically positive definite weighted matrix M (with details in 3.4), which divides the algorithms into different types.

The details of some newly designed algorithms and how they work are divided into several sub-sections below.

3.1 Classification of Algorithms

The methods could be divided into several types under different criteria: (file names consistent with the codes in the author's github)

(i) Classified **with different secant equations**:

- **Direct updates** (file names: *nmduX.m*) $Bs = y$;
- **Inverse updates** (file names: *nmiuX.m*) $Hy = s$.

(ii) Classified with **different weighted matrix** M in Frobenius norm, i.e., $\|X\|_{F,M} := \text{tr}(X^T M^{-1} X)$.

File names:

- *xxx1.m*, represent $M = I$ cases.
- *xxx2.m*, represent $M = \tau I + UU^T$ cases.
- *xxx3.m*, represent implicit cases, i.e., for some M, U can be shown to satisfy BFGS update formula in the sense of Cholesky factors. The concrete formula:

$$U^+ = U + \frac{\alpha}{u(\tau)^T s}(u(\tau) - \alpha UU^T s)s^T U \tag{8}$$

where

$$\alpha = \sqrt{\frac{u(\tau)^T s}{s^T UU^T s}},\ u(\tau) = y - \tau s,$$

$s \rightarrowtail y$ and $u(\tau) = s - \tau y$ for inverse update case.

(For file names: "*xxx*" before the number is emblematic of new method's name (direct/inverse update), and "*X*" after the letters is a number, corresponding to the weighted matrix.)

3.2 Initialization and Construction

In the first several steps, some conventional methods were applied for choosing an appropriate initial matrix U.

- For **direct update** methods: Use SR1 at initial k steps. Construct the initial low rank matrix U instantly after k steps using the projection (to symmetric positive semidefinite space $\mathbb{S}_+^k$) of the compact form of the SR1 method:

$$U = (Y_k - B_0 S_k)\Delta^{-\frac{1}{2}},$$

where

$$\begin{cases} V\Lambda V^T = D_k + L_k + L_k^T - S_k^T B_0 S_k \\ \Delta = V \max\{\Lambda, \mathbf{0}\} V^T \end{cases}.$$

- For **inverse update** methods (for even k): Use BFGS at initial $\frac{k}{2}$ steps. Construct the initial low rank matrix U instantly after $\frac{k}{2}$ steps using the projection (to symmetric positive semidefinite space $\mathbb{S}_+^k$) of the compact form of the BFGS method:

$$U = [S_k, H_0 Y_k]\Delta^{\frac{1}{2}},$$

where

$$\begin{cases} V\Lambda V^T = \begin{bmatrix} R_k^{-T}(D_k + Y_k^T H_0 Y_k)R_k^{-1} & -R_k^{-T} \\ -R_k^{-1} & \mathbf{0} \end{bmatrix} \\ \Delta = V\max\{\Lambda, \mathbf{0}\}V^T \end{cases}.$$

3.3 Choice of The Scale

The approach used to update coefficients of I in the form of B, i.e., τ_k in (6): τ^+ is the one closest to τ, but in some feasible regions, usually we choose

$$\left[a\frac{s^Ty}{s^Ts}, b\frac{s^Ty}{s^Ts}\right],\ 0 < a < b < 1, \tag{9}$$

as the feasible region (for inverse update case, interchange s and y).

3.4 Low-Rank Matrix Update Formulas

In the updating process (without implicit update) (6), derived from [2], every step U should be updated by the unique formula

$$U^+ = U + R(v^\star) = U + \left[\frac{(s^TUv^\star)Ms}{s^Tu(\tau)s^TMs} + \frac{u(\tau) - Uv^\star}{s^Tu(\tau)}\right]v^{\star T} - \frac{Mss^TU}{s^TMs}, \tag{10}$$

and the choice of M in different methods holds the same as above (details in 3.1, except for *xxx3.m*, whose updating formula follows (8)).

Note that

$$u(\tau) = \begin{cases} s - \tau y & \text{Inverse Update} \\ y - \tau s & \text{Direct Update} \end{cases}$$

At every step, $v^\star$ in (10) is the solution to

$$v^\star = \arg\min_w \left\{\psi(w) = \frac{1}{2}w^TAw + b^Tw : \|w\|_2 = \Delta\right\}, \tag{11}$$

where

$$\begin{cases} A = U^TM^{-1}U - \dfrac{U^Tss^TU}{s^TMs} \\ b = -U^TM^{-1}u(\tau) \\ \Delta = \sqrt{y^Tu(\tau)} \text{ or } \sqrt{s^Tu(\tau)} \quad \text{inverse/direct} \end{cases}.$$

There are several efficient numerical methods for solving this sub-problem.

(1) An algorithm for computing trust-region steps (Newton methods + safe-guarding), proposed by Moré and Sorensen [6]. With the translation of matrix A (in the sense of eigenvalue, i.e., $\widetilde{A} = A - \lambda_{\min}(A)I$), the algorithm for trust-region problem can be applied to above sphere constrained sub-problem.

(2) An algorithm using a hybrid of bisection and Newton, proposed by Ye [7]. The method is designed for solving quadratic functions with a sphere constraint numerically.

(3) Directly applied Newton method, with simple line search (for the sake of safeguarding).

4 Experiment Results

The experiments were designed with several cases. In every case, we compare the newly designed quasi-Newton methods with traditional L-BFGS to inspect their properties and efficiency. All methods hold the same memory storage of dimension n vectors in one comparison.

The names in the legend of all figures follow the same classfication rule as those in 3.1.

4.1 Some Simple Functions

There are some common-used simple cases for initial testing. The results (diagrams) show the efficiency of different algorithms by visualizing the change of their function value and gradient norm with iterations.

- The first test case (results in figures 1 and 2): quadratic form

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x}, \text{ where } Q \text{ is positive definite.}$$

- Another test case (results in figures 3 and 4): rank one approximation of a matrix

$$f(\boldsymbol{x}) = \frac{1}{4}\|\boldsymbol{x}\boldsymbol{x}^T - Q\|_F, \text{ where } Q \text{ has a positive eigen-value.}$$

- A third case (results in figures 5 and 6): widely-used Rosenbrock function

$$f(\boldsymbol{x}) = \sum_{i=1}^{N-1} C(x_{i+1} - x_i^2)^2 + (1 - x_i)^2.$$

4.2 Logistic Regressions

Further tests and comparisons of the new method scheme and original L-BFGS algorithm are doing on logistic regression with non-convex functions. The testing datasets come from LIBSVM Data (moderate size).

The y-axis of figures (from left to right, from top to bottom) represents function value, gradient norm, function value and step-size, respectively.

Loss/Cost functions used (minimized) in the numerical experiments are:

- Mean Square Error, Sigmoid Function:

$$f(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m}\left(\text{sigmoid}(\boldsymbol{x}_i^T\boldsymbol{w}) - y_i\right)^2,$$

where $\text{sigmoid}(z) = \dfrac{1}{1+e^{-z}}$.

- Mean Absolute Error, Hyperbolic Tangent:

$$f(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} 1 - \tanh\left(y_i \boldsymbol{x}_i^T \boldsymbol{w}\right).$$

- Modified Soft ReLU:

$$f(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} \log\left[1 + \exp\left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2\right].$$

The diagrams show the efficiency of different algorithms by visualizing the change of their function-value and gradient-norm with iterations and time (see figures 7, 8 and 9 for detailed performance).

4.3 Deep Neural Networks

The large-scale functions used in our testing come from the autoencoder (a special type of artificial neural network), which is widely used in feature engineering. Define a reference probability distribution and let $\boldsymbol{x} \sim \boldsymbol{\mu}_{\text{ref}}$. Loss function here becomes

$$\mathbb{E}_{\boldsymbol{x}}\left[d\left(\boldsymbol{x}, D_{\boldsymbol{\theta}} \circ E_{\boldsymbol{\phi}}(\boldsymbol{x})\right)\right],$$

where $(\mathcal{X}, d)$ is a metric space, $E_{\boldsymbol{\phi}}$ is the encoder (constructed by a deep neural network, parameter $\boldsymbol{\phi}$) and $D_{\boldsymbol{\theta}}$ is the decoder (constructed by another deep neural network, parameter $\boldsymbol{\theta}$). After sampling, we can use deterministic optimization methods to train the autoencoder by solving

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} d\left(\boldsymbol{x}_i, D_{\boldsymbol{\theta}} \circ E_{\boldsymbol{\phi}}(\boldsymbol{x}_i)\right)$$

Two datasets (i.e., different spaces $(\mathcal{X}, d)$), MNIST and CIFAR10, were applied in the testing cases, respectively.

In the figures, different names in the legend represent different quasi-Newton methods, while the prefixes display line search methods. The same prefix means all methods used the same line search (see figures 10 and 11 for detailed performance).

4.4 Discussions on Experiment Results

General Convegence Rate and Efficiency

In general, under the same memory storage of vectors containing second-order information, our new methods performed competitively with L-BFGS in algorithm efficiency. In small, simple cases or large, non-convex cases, L-BFGS and new quasi-Newton methods have comparable convergence rates with subtle differences in different question types. Hence, in future studies, these new methods can be considered an alternative to L-BFGS in large-scale optimizations.

Concretely, the convergence behaviors of different new methods in our numerical experiments follow that:

- direct update with weighted matrix I (i.e., *nmdu1*): converges faster and behaves explicitly better than L-BFGS in the quadratic form case, and some logistic regression cases.

- direct update with implicit weighted matrix case (i.e., *nmdu3*): competitive with L-BFGS in some cases, but does not show good enough convergence rates in our test cases

- inverse update with weighted matrix I (i.e., *nmiu1*): competetitive with L-BFGS in most cases, and beat L-BFGS in some case like sparse Q in quadratic form and some logistic regression cases.
- inverse update with weighted matrix $\tau I + UU^T$ (i.e., *nmiu2*): works the best among new methods in the numerical tests we have done, competetitive with L-BFGS in most cases, and beat L-BFGS in some case like sparse Q in quadratic form, rosenbrock functions and some logistic regression cases.
- inverse update with implicit weighted matrix (i.e., *nmiu3*): behaviors similar to *nmiu2*, competetitive with L-BFGS in most cases, and less time-consuming than *nmiu2* due to fewer computational costs in every step.

Furthermore, since L-BFGS and the new quasi-Newton methods have different updating approaches, it is reasonable to believe that they should have advantages in different fields. We will continue doing other numerical tests to discover more new algorithms' properties.

Stability of Direct and Inverse Updates

The inverse updates exhibit more stability from all the numerical results than the direct updates. Since the inverse updates worked in all cases and converged well most of the time (competitive with L-BFGS), especially the inverse update with the last approximated Hessian as the weighted matrix (a.k.a., nmiu2). In contrast, direct updates fail to work (diverge or behave slowly) in some cases like rosenbrock functions and autoencoder questions.

Nevertheless, the direct updates worked well in some cases. For instance, in the cases of quadratic form and the regression under some datasets, particularly for the direct update with identity weighted matrix y (a.k.a., nmdu1). Hence, direct updates are still worthwhile in some fields of the actual industry.

Acknowledgements

The author extends sincere gratitude to Professor Yin Zhang for his initial conception, patient guidance during the research, and for leading the author to learn how to do research at the beginning of his study. The author also thanks Hanfeng Zeng for his help designing large numerical tests, and Professor Yinyu Ye for his helpful discussions on his efficient algorithm for numerically solving the sub-problem.

The research project was supported by URA Support in CUHKSz.

Appendices

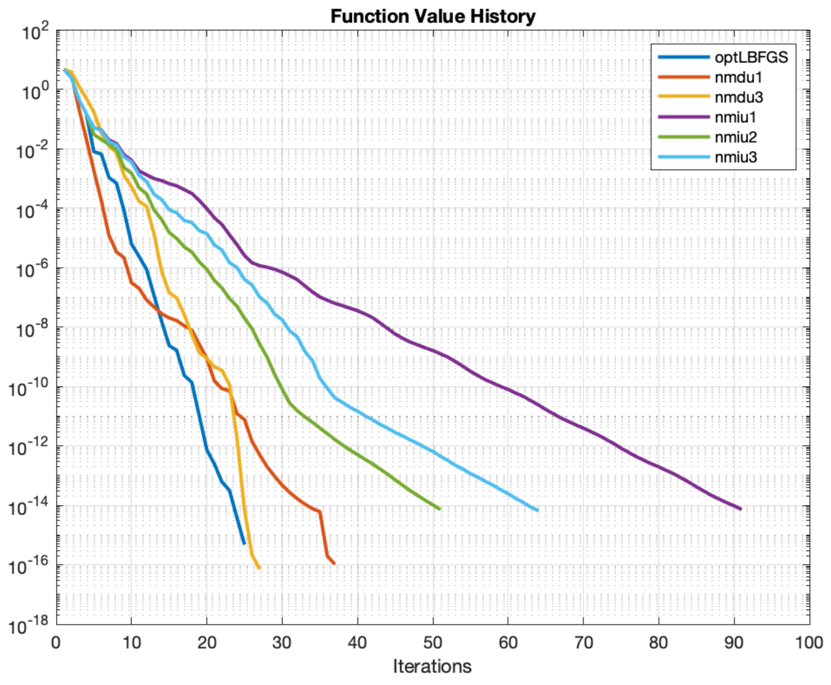
A

Pseudo-Algorithm of New Method Scheme

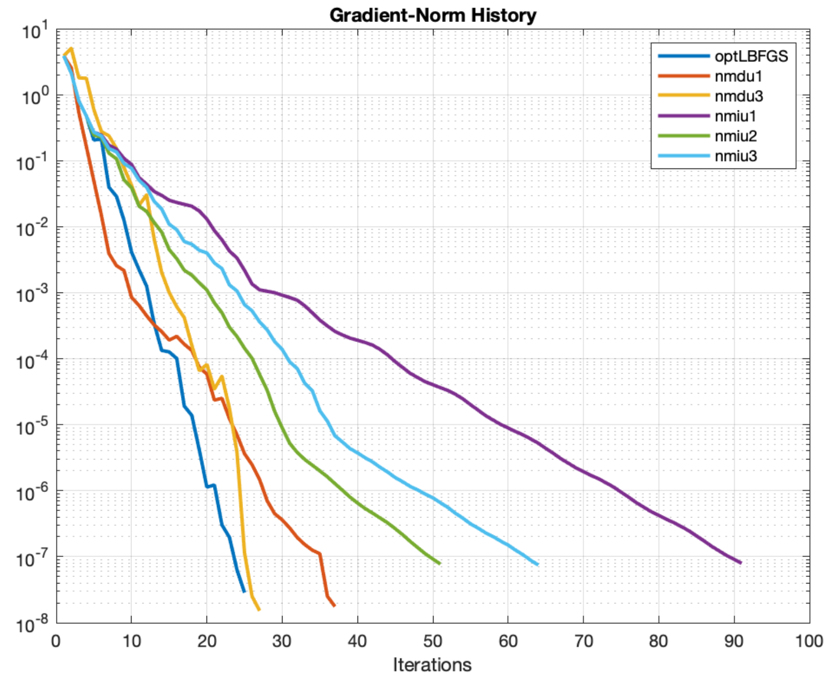
Here lists the pseudo-algorithms for our new method scheme.

Algorithm 1: Quasi-Newton New Scheme

```
Data: Initial point: $\boldsymbol{x}_0$; Function Structure: myFunc; Gradient Tolerance: gtol; Maximum Iteration:
      maxIter; Memory Size: mS
% Step I: Initialize Low-Rank Matrix $U$
% Inverse Updates:  $target = mS/2$, BFGS-update, BFGS inverse compact
  form
% Direct Update 1:  $target = mS$, SR1-update, SR1 compact form
% Direct Update 3:  $target = 1$, construct $U$ with a feasible $v^\star$
  directly
1  for $iter = 1 : target$ do
2      $s \longleftarrow x - x_{\text{last}}$;
3      $y \longleftarrow g - g_{\text{last}}$;
4      $S \longleftarrow [S, s]$;
5      $Y \longleftarrow [Y, y]$;
6      $d \longleftarrow$ FindSearchDirection $(S, Y, g)$ ;          % under different methods
7      if $iter = target$ then
8          $U \longleftarrow$ Construction $(S, Y)$ ;       % different methods with different
           constructions, details see above
% Step II: Update under New Quasi-Newton Scheme
9  for $iter = target + 1 : maxIter$ do
10     $s \longleftarrow x - x_{\text{last}}$;
11     $y \longleftarrow g - g_{\text{last}}$;
12     $\tau \longleftarrow$ ChooseScale $(s, y, \tau_{\text{last}})$ ;                    % with some criteria
13     $U \longleftarrow$ UpdateFormula $(U, s, y)$ ;          % under different update schemes
14     $d \longleftarrow$ FindSearchDirection $(\tau, U, g)$ ;  % direct or inverse multiplication of
       $\tau I + UU^T$ and $-g$
15     $\alpha \longleftarrow$ LineSearch $(myFunc, d, x)$;
16     $x \longleftarrow x + \alpha d$ ;
17     Update $f, g, \|g\|$ correspondingly;
18     if $\|g\| \leq gtol\|g_{init}\|$ then
19         Output $x, myFunc(x)$;
20         exit;
```

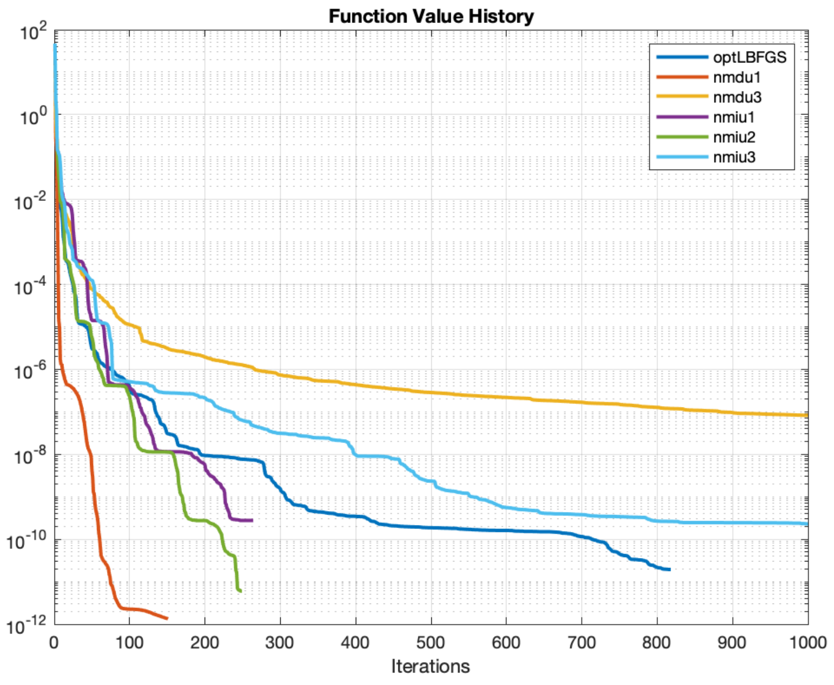


(a) Sparse Positive definite Q, Function Value

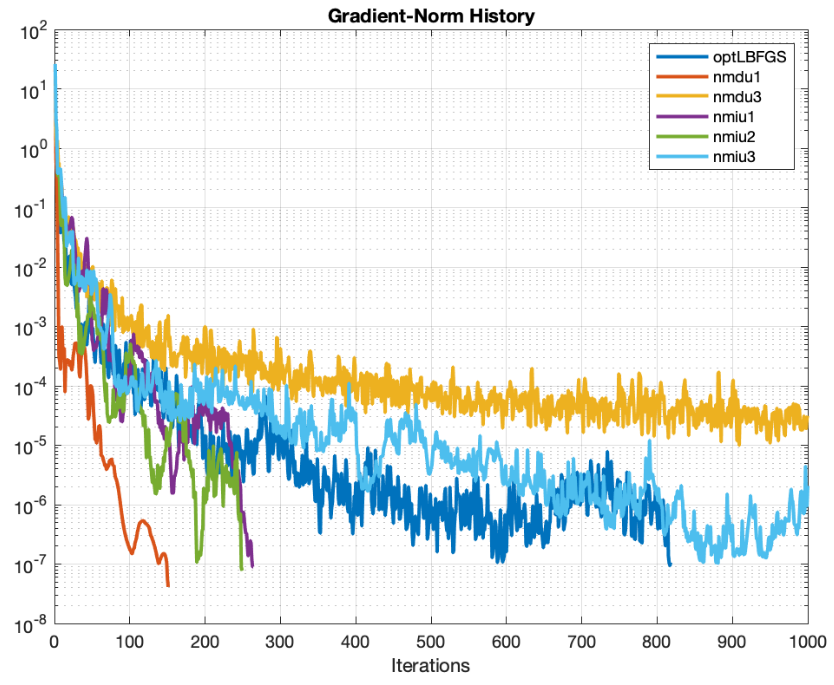


(b) Sparse Positive definite Q, Gradient Norm

Figure 1: Case I-1: Quadratic Form, Dense Positive definite Q

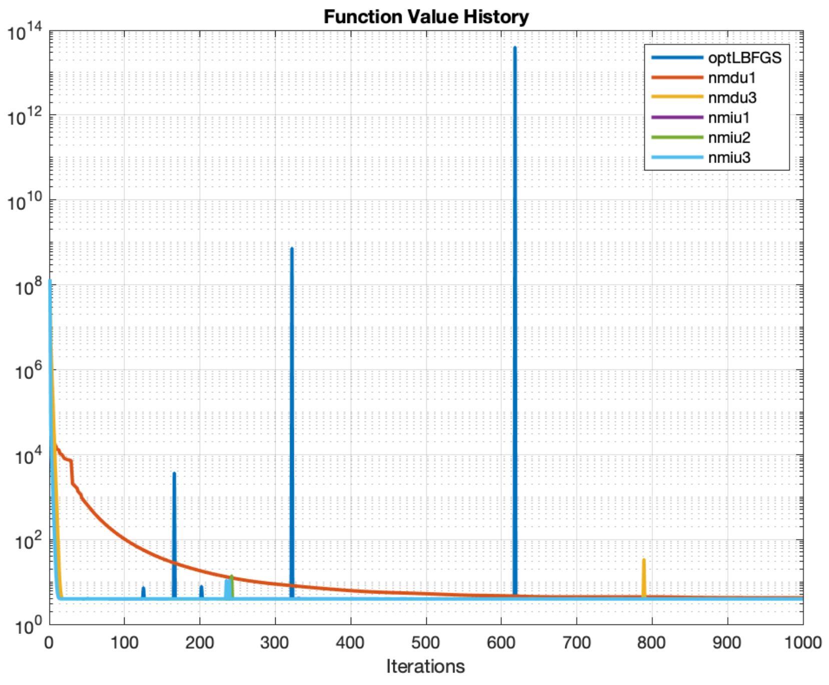


(a) Function Value

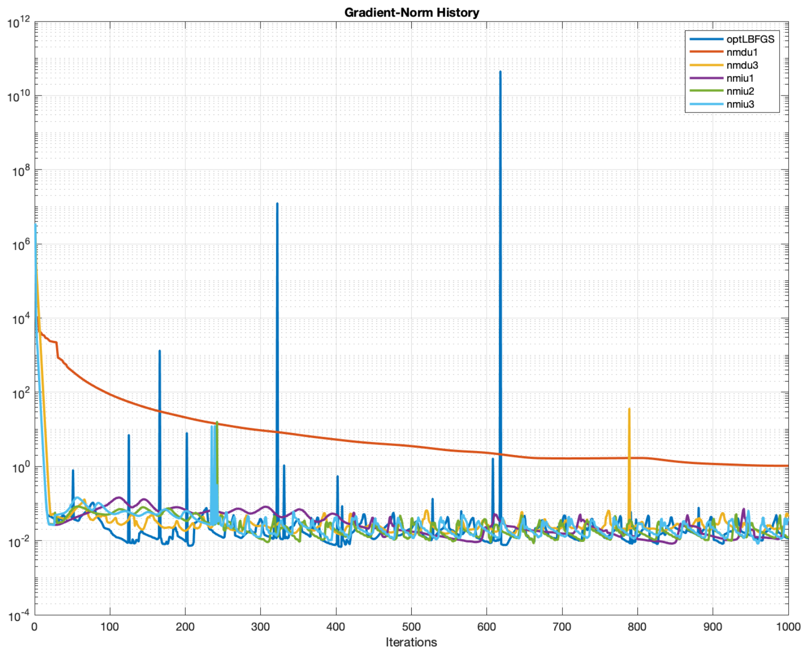


(b) Gradient Norm

Figure 2: Case I-2: Quadratic Form, Dense Positive definite Q

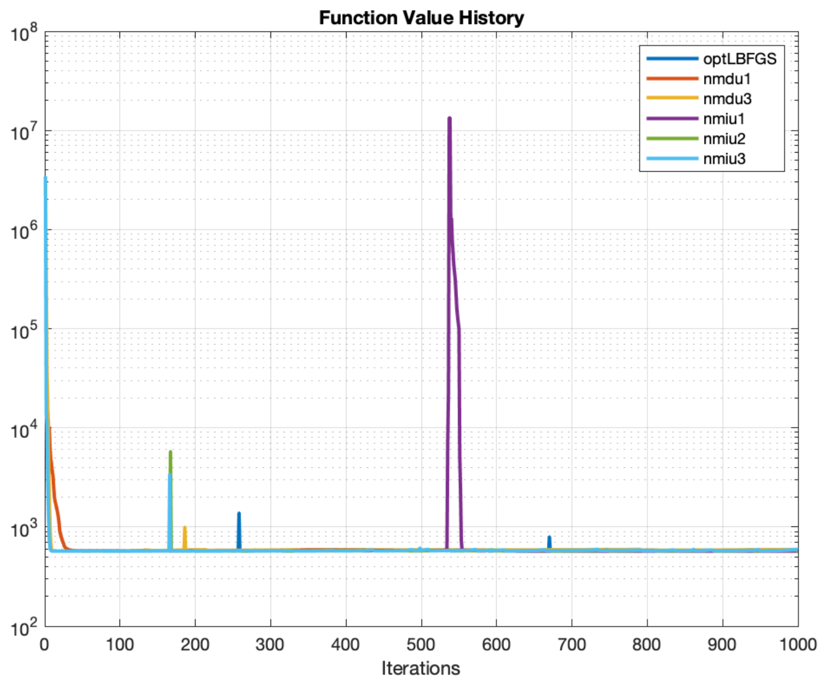


(a) Function Value

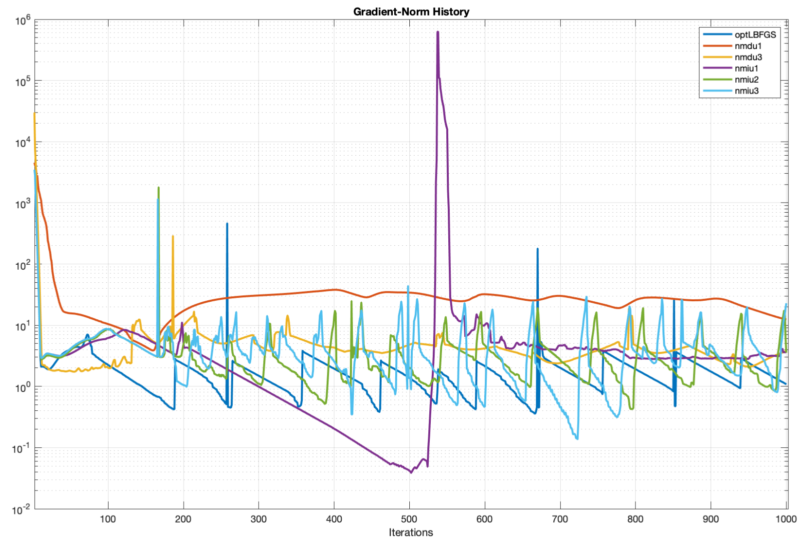


(b) Gradient Norm

Figure 3: Case II-1: Rank One Approximation, Sparse Ill-Conditioned Q

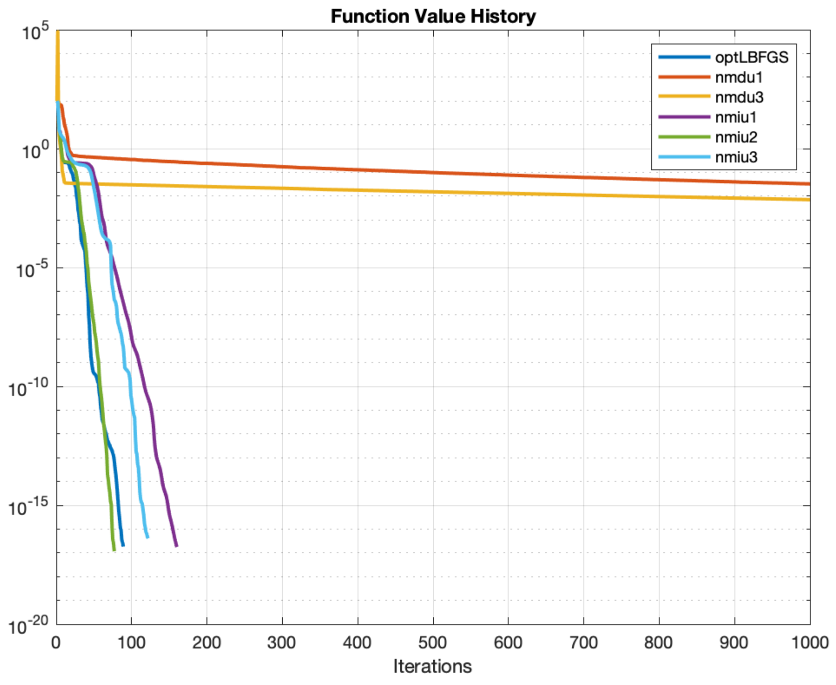


(a) Function Value

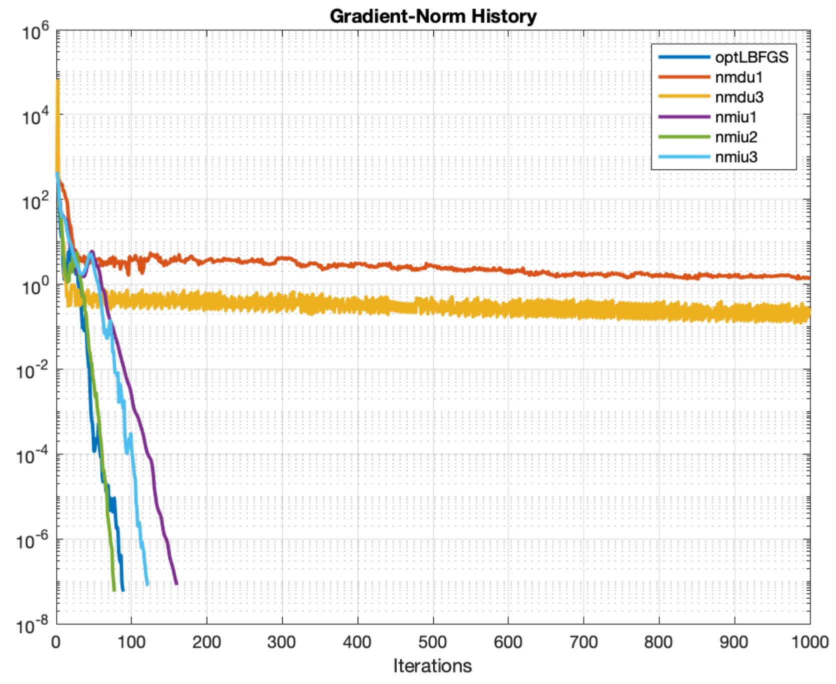


(b) Gradient Norm

Figure 4: Case II-2: Rank One Approximation, Lower Hessenberg Q

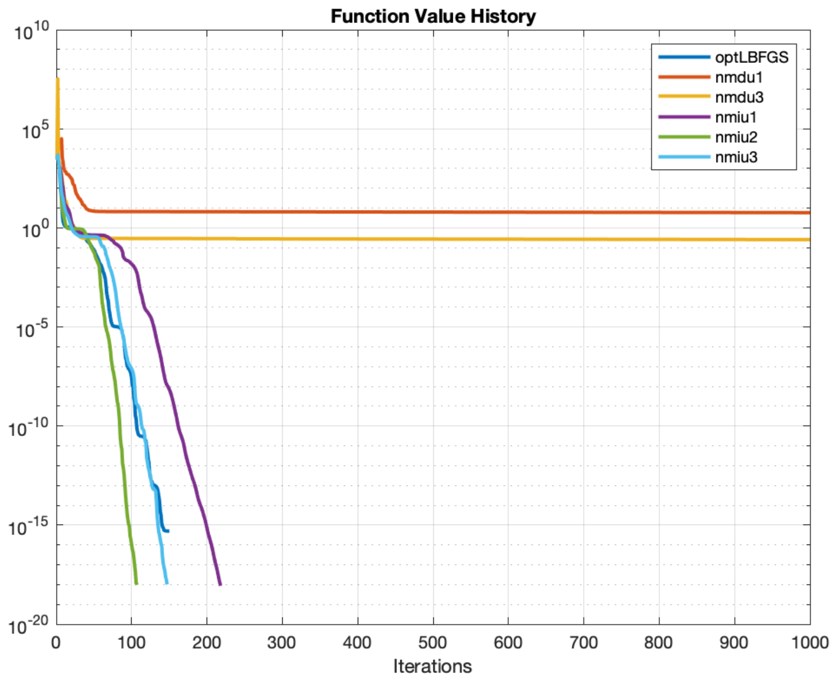


(a) Function Value

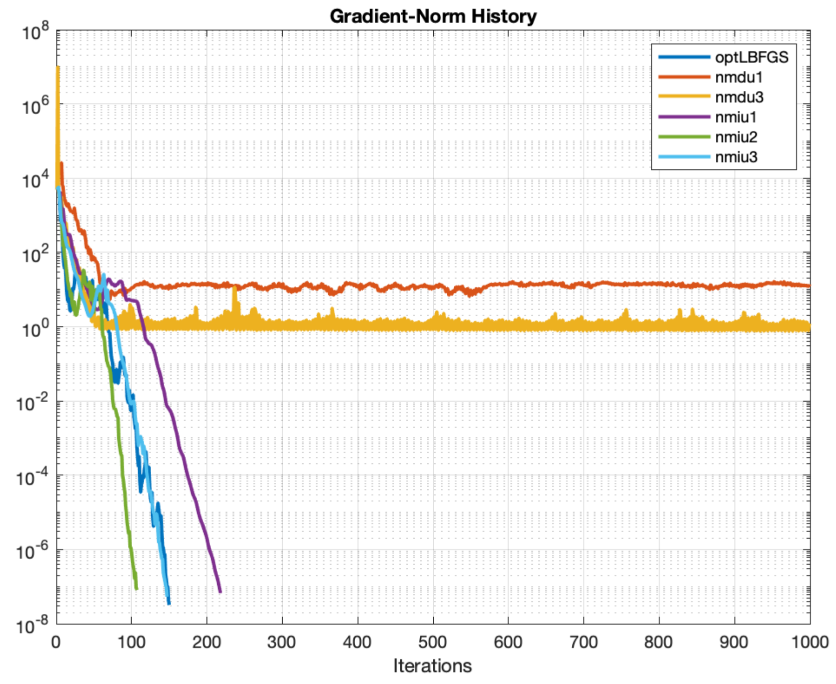


(b) Gradient Norm

Figure 5: Case III-1: Rosenbrock, $C = 100$, $N = 2000$



(a) Function Value



(b) Gradient Norm

Figure 6: Case III-2: Rosenbrock, $C = 1000, N = 2000$

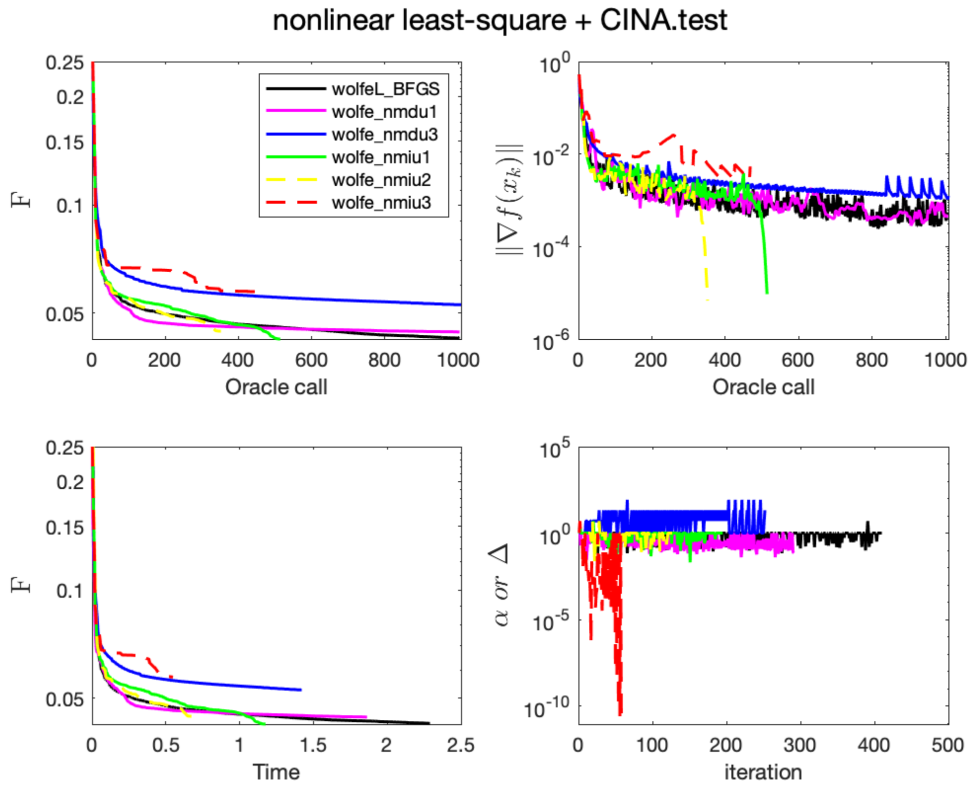


Figure 7: Loss Function: Sigmoid; Dataset: CINA.test

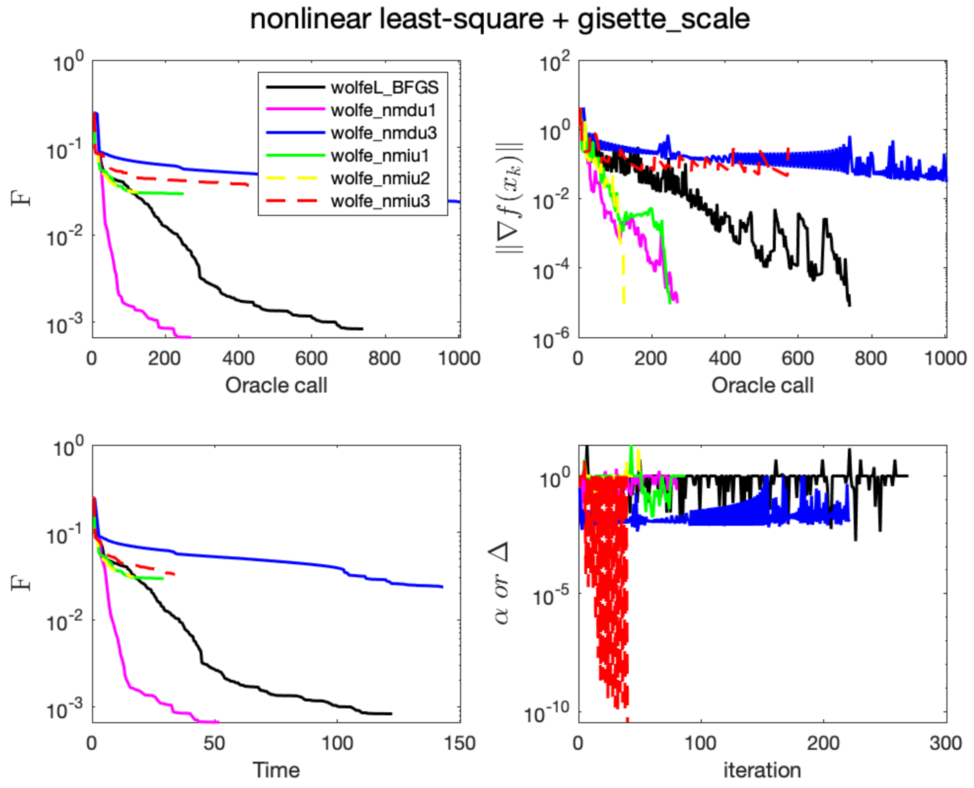


Figure 8: Loss Function: Sigmoid; Dataset: gisette_scale

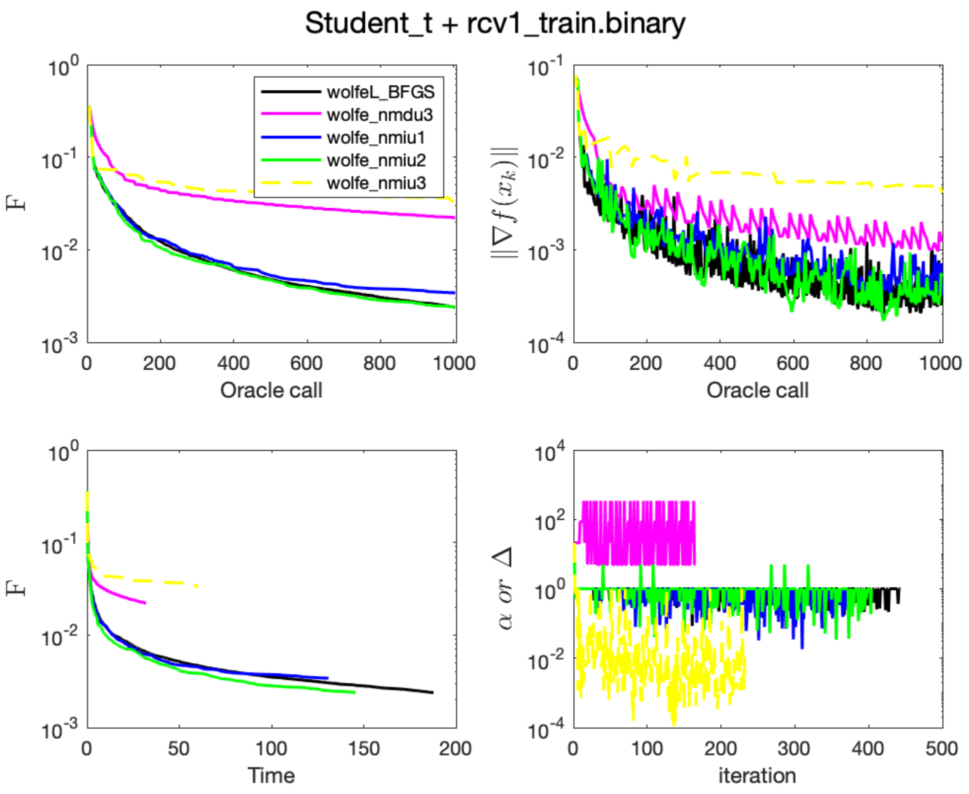
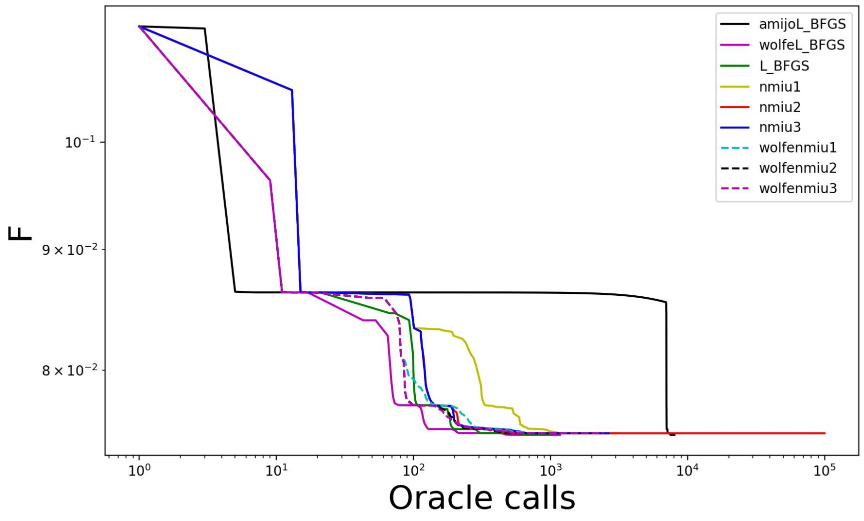


Figure 9: Loss Function: Modified Soft ReLU; Dataset: rcv1_train.binary



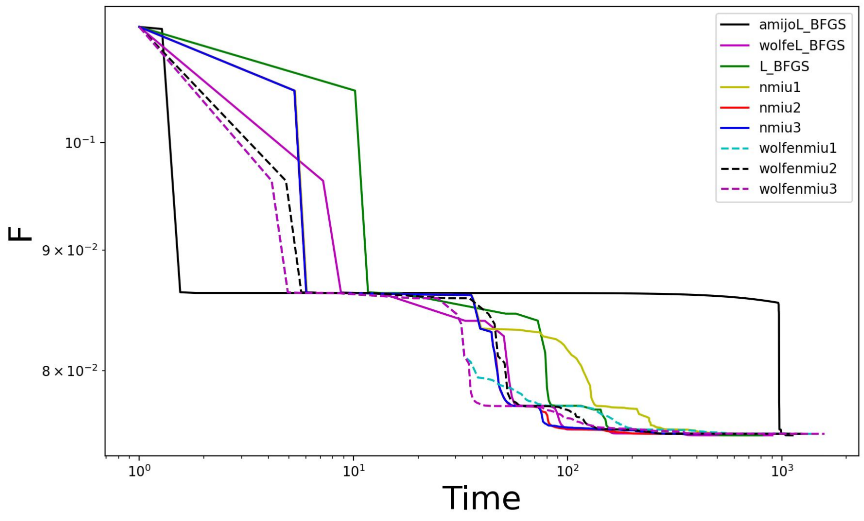
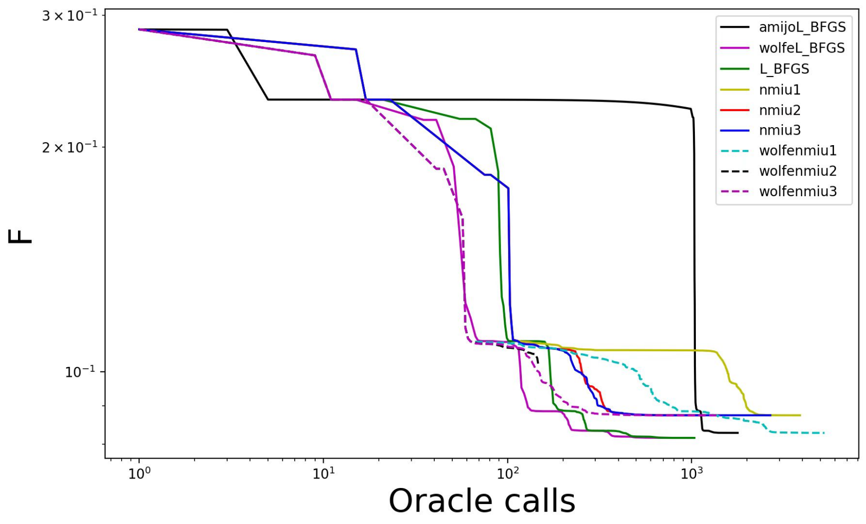


Figure 10: Autoencoder: MNIST



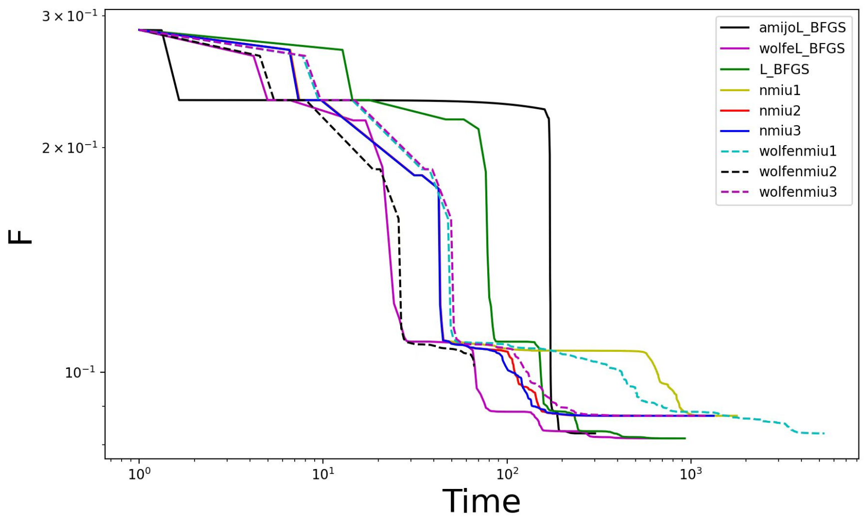


Figure 11: Autoencoder: CIFAR10

B References

[1] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, 2nd ed., Springer, NY, 2006.

[2] Y. ZHANG AND R. P. TEWARSON, *Least-change updates to Cholesky factors subject to the nonlinear quasi-Newton condition*, IMA J. Numer. Anal., 7(1987), pp. 509-521.

[3] Y. ZHANG AND R. P. TEWARSON, *Quasi-Newton algorithms with updates from the preconvex part of Broyden's family*, IMA J. Numer. Anal., 8(1988), pp. 487-509.

[4] D. C. LIU AND J. NOCEDAL, *On the limited memory BFGS method for large scale optimization*. Math. Prog., 45(1989), pp. 503-528.

[5] T. LIU, S. BOUAZIZ AND L. KAVAN, *Quasi-newton methods for real-time simulation of hyperelastic materials*. Acm. Trans. Graps., 36(2017), pp. 1-16.

[6] J. J. MORÉ AND D. C. SORENSEN., *Computing a trust region step*, SIAM J. Stat. Comput., 4, 3(1983), pp. 553-572.

[7] Y. YE., *A new complexity result minimization of a quadratic function with a sphere constraint*, Re. Advan. Glob. Optim., (1992), pp. 19-31.