Constrained Optimization via Exact Augmented Lagrangian and Randomized Iterative Sketching

Ilgee Hong [1 *] **Sen Na** [2 3 *] **Michael W. Mahoney** [2 3 4] **Mladen Kolar** [5]

Abstract

We consider solving equality-constrained nonlinear, nonconvex optimization problems. This class of problems appears widely in a variety of applications in machine learning and engineering, ranging from constrained deep neural networks, to optimal control, to PDE-constrained optimization. We develop an adaptive inexact Newton method for this problem class. In each iteration, we solve the Lagrangian Newton system inexactly via a *randomized iterative sketching* solver, and select a suitable stepsize by performing line search on an *exact augmented Lagrangian* merit function. The randomized solvers have advantages over deterministic linear system solvers by significantly reducing per-iteration flops complexity and storage cost, when equipped with suitable sketching matrices. Our method adaptively controls the accuracy of the randomized solver and the penalty parameters of the exact augmented Lagrangian, to ensure that the inexact Newton direction is a descent direction of the exact augmented Lagrangian. This allows us to establish a *global almost sure convergence*. We also show that a unit stepsize is admissible locally, so that our method exhibits a *local linear convergence*. Furthermore, we prove that the linear convergence can be strengthened to *superlinear convergence* if we gradually sharpen the adaptive accuracy condition on the randomized solver. We demonstrate the superior performance of our method on benchmark nonlinear problems in CUTEst test set, constrained logistic regression with data from LIBSVM, and a PDE-constrained problem.

[*]Equal contribution [1]Department of Statistics, University of Chicago [2]Department of Statistics, University of California, Berkeley [3]International Computer Science Institute [4]Lawrence Berkeley National Laboratory [5]Booth School of Business, University of Chicago. Correspondence to: Sen Na <senna@berkeley.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

We study constrained optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) \qquad \text{s.t.} \;\; c(\boldsymbol{x}) = \mathbf{0}, \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is the objective function, $c:\mathbb{R}^n\to\mathbb{R}^m$ is the equality constraint function, and both of them are nonlinear, possibly nonconvex, and twice continuously differentiable. Problem (1) appears widely (as a subproblem) in a variety of applications, including constrained deep neural networks (Chen et al., 2018), physical informed neural networks (Cuomo et al., 2022), PDE-constrained optimization (Kouri et al., 2014), optimal control (Lewis et al., 2012), and constrained model estimations (Scott & Tůma, 2022). There exist numerous methods for solving Problem (1), such as projected first- and second-order methods, penalty methods, augmented Lagrangian methods, and sequential quadratic programming (SQP) methods (Nocedal & Wright, 2006). Among these methods, SQP is one of the most effective methods for both small and large problems. Compared to projected methods, SQP does not involve a projection step that is generally expensive for nonlinear equality constraints. Compared to penalty methods, SQP does not regularize the objective that destroys the structure of the problem and suffer from ill-conditioning issue (Krishnapriyan et al., 2021). Compared to augmented Lagrangian methods, SQP is more robust to the (dual) initialization (Curtis et al., 2014a).

With only equality constraints, SQP is equivalent to an application of Newton method to the Karush–Kuhn–Tucker (KKT) conditions (i.e., first-order optimality conditions). As the second-order method, SQP is (not surprisingly) very efficient, taking only a few iterations to find a local solution to the problem; while, at the same time, it can struggle to solve the Newton system (of the Lagrangian) efficiently at each step. Solving Newton systems dominates the computational cost of Newton methods, and deriving exact Newton directions is often prohibitive for large-scale problems.

1.1. Main Results

Motivated by the above bottleneck, we propose an adaptive inexact Newton (SQP) method for Problem (1) that applies randomized iterative sketching solvers to solve Newton systems inexactly. Our method is called **AdaSketch-Newton**. At each step, AdaSketch-Newton solves the Lagrangian New-

ton system by leveraging the *sketch-and-project* framework, which was originally developed in Gower & Richtárik (2015) and further investigated for various (unconstrained) optimization problems (Luo et al., 2016; Doikov & Richtárik, 2018; Gower et al., 2019; Mutny et al., 2020; Derezinski et al., 2020b; Na & Mahoney, 2022; Dereziński & Rebrova, 2022). The framework unifies popular randomized iterative linear system solvers, such as the randomized (block) coordinate descent and randomized (block) Kaczmarz method. Compared to deterministic solvers, randomized solvers require less per-iteration flops complexity and storage cost when using suitable sketching matrices (e.g., a sparse sketching matrix). See Murray et al. (2023) for recent software development. However, using inexact randomized solver leads to a much more involved algorithm design as well as a more challenging analysis, since the approximation error of the solution is random and does not monotonically decrease as more iterations of the solver are performed. As we review in Section 1.2, existing inexact Newton methods with randomized sketching solvers are designed for unconstrained problems; we provide the first inexact Newton method with a randomized solver for constrained problems.

In the proposed AdaSketch-Newton algorithm, we use an *exact augmented Lagrangian merit function*, taking the form

$$\mathcal{L}_{\boldsymbol{\eta}}(\boldsymbol{x},\boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda}) + \frac{\eta_1}{2}\|c(\boldsymbol{x})\|^2 + \frac{\eta_2}{2}\|\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})\|^2, \quad (2)$$

where $\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T c(\boldsymbol{x})$ is the Lagrangian function of (1), with $\boldsymbol{\lambda} \in \mathbb{R}^m$ being the Lagrangian multipliers and $\boldsymbol{\eta} = (\eta_1, \eta_2)$ being the penalty parameters. By "exact," we mean that, compared with standard augmented Lagrangian, it has an additional penalty term $\eta_2\|\nabla_{\boldsymbol{x}}\mathcal{L}\|^2/2$ that biases the optimality error. With this additional term, one can show that the solution of the unconstrained problem $\min_{\boldsymbol{x},\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}}$ is also the solution of (1) and vice versa, provided that $\boldsymbol{\eta}$ are suitably specified. By "merit function," we mean (2) is (only) used to determine whether or not a new iterate approaches toward a local solution to Problem (1). In other words, (2) does not affect the computation of inexact Newton direction (thus, our method does not suffer ill-conditioning issue caused by $\boldsymbol{\eta}$); but given a direction, we select a suitable stepsize by performing line search on (2). The merit function plays a crucial role in constrained optimization, since the objective function f alone is not suitable to justify the direction (the step that decreases f may severely violate the constraint c). AdaSketch-Newton adaptively controls the accuracy of the randomized sketching solver and selects suitable penalty parameters $\boldsymbol{\eta}$ to ensure that the inexact Newton direction is a descent direction of (2). Our adaptive design balances the number of global outer loop iterations and inner loop iterations of the solver, while at the same time securing a fast local convergence. By virtue of the adaptivity, all the input parameters of the algorithm are problem independent.

Under mild assumptions, we show that AdaSketch-Newton enjoys a global almost sure convergence — starting from any initial point, the KKT residual converges to zero. We also show that our adaptive design ensures a unit stepsize to be admissible locally, and establish a local linear convergence rate. Furthermore, we prove that the linear rate can be strengthened to a superlinear rate, as long as we gradually sharpen the adaptive accuracy condition on the randomized solver, which is simply controlled by an input tuning parameter. As we review in Section 1.2, the local rate of inexact Newton methods is mostly investigated for unconstrained problems (Gower et al., 2019; 2021; Hanzely et al., 2020; Dereziński & Rebrova, 2022; Yuan et al., 2022), while is largely missing in constrained cases (Byrd et al., 2008; 2010; Curtis et al., 2014b; Gu et al., 2017; Burke et al., 2020).

We implement AdaSketch-Newton and benchmark it against constrained nonlinear problems in CUTEst test set (Gould et al., 2014), constrained logistic regression with datasets from LIBSVM (Chang & Lin, 2011), and a PDE-constrained problem (Hintermüller et al., 2002). We show the superior performance of our method, in terms of both accuracy and efficiency, by comparing it with a prominent inexact Newton method that employs a deterministic solver and ℓ_1 merit function (Byrd et al., 2008) and standard augmented Lagrangian method (Nocedal & Wright, 2006). We also show our method is robust to tuning parameters.

1.2. Related Literature

Our method relates to several inexact Newton-type methods in the literature that we briefly review below. We divide the review into unconstrained and constrained optimization.

Unconstrained optimization. The majority of inexact Newton methods are designed for unconstrained problems. For example, some inexact Newton methods via randomized iterative sketching have been proposed, including Stochastic Dual Newton Ascent (SDNA) (Qu et al., 2016), Sketched Online Newton (SON) (Luo et al., 2016), Stochastic Subspace Cubic Newton (SSCN) (Hanzely et al., 2020), Randomized Block Cubic Newton (RBCN) (Doikov & Richtárik, 2018), Randomized Subspace Newton (RSN) (Gower et al., 2019), and Randomized Subspace Regularized Newton (RSRN) (Fuji et al., 2022). These methods incorporate sketching techniques into classical Newton methods or cubic regularized Newton methods. Instead of solving the original large-scale subproblem in each iteration, the methods solve a sketched small-scale subproblem, and it is proved that the sketched direction decreases the objective, due to the convexity assumption and/or unconstrained nature of the problem. With a good understanding on the convergence of sketching solvers (Gower et al., 2021; Dereziński & Rebrova, 2022), the (local) convergence rates of these methods are also established. Empirical results

illustrate that randomized Newton methods integrate the benefits of randomization and the efficiency of second-order methods. The proposed AdaSketch-Newton is the first generalization of the aforementioned methods to constrained nonlinear problems (1). Our method adaptively controls the accuracy of the sketching solver and the penalty parameters of the merit function, and takes the constraint violation into account when showing the descent property of the sketched inexact Newton direction.

There are numerous Newton methods where subproblems can be solved inexactly by deterministic solvers such as MINRES and CG (Fong & Saunders, 2012). For example, different variants of Newton-CG methods have been developed with line search (Royer & Wright, 2018; Royer et al., 2019; Bollapragada et al., 2018; Yao et al., 2022), trust region (Curtis et al., 2021a), and cubic regularization (Curtis et al., 2018; Yao et al., 2021). Newton-MR has also recently been reported and analyzed (Liu & Roosta, 2022; Roosta et al., 2022). Another line of work focuses on constructing cheap stochastic Hessian approximations by either sketching or subsampling (Pilanci & Wainwright, 2017; Roosta-Khorasani & Mahoney, 2018; Derezinski et al., 2020a; Lacotte et al., 2021; Derezinski et al., 2021; Na et al., 2022b). We refer to Berahas et al. (2020) for a comparison of these methods. For some of the methods (e.g., Roosta-Khorasani & Mahoney (2018)), the resulting Newton systems can also be solved inexactly via deterministic solvers. However, the convergence guarantees of those methods hold in high probability or expectation and (more or less) rely on the convexity of problems. This differs from our almost sure convergence guarantee for nonlinear problems.

Constrained optimization. To our knowledge, there are only a handful of inexact Newton methods for constrained problems, which all employ deterministic solvers for solving Newton systems and are often called inexact SQP methods. Byrd et al. (2008; 2010) proposed two inexact methods for Problem (1), which adopt nonsmooth merit functions in line search step. The methods bound the residuals of the solver by a few *fixed* tuning parameters, and are then refined and adapted to incorporate inequality constraints (Curtis et al., 2014b; Burke et al., 2020). Recently, stochastic SQP methods have been reported (Berahas et al., 2021a;b; Curtis et al., 2021b; Na et al., 2023; 2022a; Fang et al., 2022; Na & Mahoney, 2022). The stepsize in most of these methods is controlled by prespecified decaying sequences (i.e., the methods are designed under the stochastic approximation regime), with the only exceptions being Na et al. (2023) and Na et al. (2022a), which adopt line search to make the methods more adaptive. The aforementioned methods established global convergence, while local rates of the methods are largely missing. Our method is the first design that incorporates the sketching technique into SQP, adaptively selects suitable stepsizes by line search, and exhibits global and local linear/superlinear convergence as exact SQP/Newton methods do.

Notation. We use $\|\cdot\|$ to denote the ℓ_2 norm for vectors and the operator norm for matrices. For any matrix $A \in \mathbb{R}^{p\times q}$, $row(A)$ denotes the row space of A and $col(A)$ denotes the column space of A. We let $H(\boldsymbol{x},\boldsymbol{\lambda}) = \nabla^2_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda}) \in \mathbb{R}^{(n+m)\times(n+m)}$ be the Lagrangian Hessian with respect to $\boldsymbol{x}$ and $G(\boldsymbol{x}) = \nabla^T c(\boldsymbol{x}) \in \mathbb{R}^{m\times n}$ be the constraint Jacobian. At the k-th iteration, we let $f_k = f(\boldsymbol{x}_k)$, $H_k = H(\boldsymbol{x}_k,\boldsymbol{\lambda}_k)$, etc., to ease the notation. For any $a, b \in \mathbb{R}$, we let $a \vee b = \max\{a, b\}$ and $a \wedge b = \min\{a, b\}$.

2. AdaSketch-Newton

We now introduce AdaSketch-Newton algorithm. For Problem (1), under certain constraint qualifications, a necessary condition for $\boldsymbol{x}^\star$ being a local solution is to have dual multipliers $\boldsymbol{\lambda}^\star$ such that

$$\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)\end{pmatrix} = \begin{pmatrix}\nabla f(\boldsymbol{x}^\star) + G(\boldsymbol{x}^\star)^T\boldsymbol{\lambda}^\star\\ c(\boldsymbol{x}^\star)\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ \mathbf{0}\end{pmatrix}. \quad (3)$$

The starting point of our algorithm is to apply the Newton method to (3). At each iteration k, we consider solving the following Newton system of the Lagrangian

$$\Gamma_k \Delta \boldsymbol{z}_k = -\nabla\mathcal{L}_k, \quad (4)$$

where

$$\Gamma_k = \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix},\ \Delta\boldsymbol{z}_k = \begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix},\ \nabla\mathcal{L}_k = \begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_k\\ c_k\end{pmatrix}.$$

Here, Γ_k approximates the Lagrangian Hessian $\nabla^2\mathcal{L}_k$ with B_k being a modification of H_k to ensure (4) is solvable (cf. Assumption 3.2). We note that solving the Newton system (4) is equivalent to solving a constrained quadratic program

$$\begin{aligned}\min_{\Delta\boldsymbol{x}}\ & \frac{1}{2}\Delta\boldsymbol{x}^T B_k \Delta\boldsymbol{x} + \nabla f_k^T \Delta\boldsymbol{x},\\ \text{s.t.}\ & c_k + G_k\Delta\boldsymbol{x} = \mathbf{0}.\end{aligned}$$

The objective is a quadratic approximation of f with Hessian coming from the Lagrangian function, while the constraint is a linear approximation of c. Instead of solving (4) exactly and having $\Delta\boldsymbol{z}_k$, we apply a randomized iterative sketching solver on (4) and derive an (random) inexact solution $\tilde{\Delta}\boldsymbol{z}_k$.

2.1. Randomized Iterative Sketching

For every *outer iteration* k, we run multiple *inner iterations* of the sketching solver, indexed by j. The solver will stop at some j and output $\tilde{\Delta}\boldsymbol{z}_k := \tilde{\Delta}\boldsymbol{z}_{k,j} = (\tilde{\Delta}\boldsymbol{x}_{k,j}, \tilde{\Delta}\boldsymbol{\lambda}_{k,j})$ as the inexact direction, once the step $\tilde{\Delta}\boldsymbol{z}_{k,j}$ triggers an adaptive step acceptance condition (introduced in Section 2.2). As we will prove in Lemmas 3.6 and 3.7, the step acceptance condition is always triggered with a finite j (with probability one).

We now provide details of the sketch-and-project framework. With different sketching matrices, the framework recovers different randomized methods including, randomized Kaczmarz method, randomized coordinate descent, etc. (Gower & Richtárik, 2015; Strohmer & Vershynin, 2008; Richtárik & Takáč, 2014). Let $S \in \mathbb{R}^{(n+m)\times d} \sim \mathcal{P}$ be a random sketching matrix following the distribution $\mathcal{P}$ (its sketching dimension d can also be random). For each outer iteration k and inner iteration j, we generate a copy of S denoted by $S_{k,j}$, and solve the following sketched Newton system

$$S_{k,j}^T \Gamma_k \boldsymbol{u} = -S_{k,j}^T \nabla \mathcal{L}_k. \tag{5}$$

Since (5) has multiple solutions including the exact direction $\Delta \boldsymbol{z}_k$, the j-th iteration of the solver selects the step $\tilde{\Delta}\boldsymbol{z}_{k,j+1}$ to be the one that is closest to the current step $\tilde{\Delta}\boldsymbol{z}_{k,j}$, i.e.,

$$\tilde{\Delta}\boldsymbol{z}_{k,j+1} = \arg\min_{\boldsymbol{u}} \|\boldsymbol{u} - \tilde{\Delta}\boldsymbol{z}_{k,j}\|^2, \quad \text{subject to (5)}.$$

An explicit updating rule is given by

$$\tilde{\Delta}\boldsymbol{z}_{k,j+1} = \tilde{\Delta}\boldsymbol{z}_{k,j} - \Gamma_k S_{k,j}(S_{k,j}^T \Gamma_k^2 S_{k,j})^\dagger S_{k,j}^T \boldsymbol{r}_{k,j}, \tag{6}$$

where $(\cdot)^\dagger$ is the Moore–Penrose pseudoinverse and $\boldsymbol{r}_{k,j}$ is the residual of $\tilde{\Delta}\boldsymbol{z}_{k,j}$, defined as

$$\boldsymbol{r}_{k,j} = \Gamma_k \tilde{\Delta}\boldsymbol{z}_{k,j} + \nabla \mathcal{L}_k. \tag{7}$$

We initialize the solver with $\tilde{\Delta}\boldsymbol{z}_{k,0} = \mathbf{0}$, and below we introduce when we should stop the inner iteration (6).

2.2. Adaptive Step Acceptance Condition

Our step acceptance condition consists of two subconditions: (i) adaptive accuracy condition enforced on the residual $\boldsymbol{r}_{k,j}$ (cf. (10)); and (ii) descent direction condition enforced on the step $\tilde{\Delta}\boldsymbol{z}_{k,j}$ (cf. (11)). Both conditions rely on the penalty parameters $\boldsymbol{\eta}_k = (\eta_{1,k}, \eta_{2,k})$ of the augmented Lagrangian (2), which we also adaptively choose. Once the step acceptance condition is triggered, we derive an inexact direction $\tilde{\Delta}\boldsymbol{z}_k = \tilde{\Delta}\boldsymbol{z}_{k,j}$ with the chosen penalty parameters $\boldsymbol{\eta}_k$. With these chosen quantities, we do line search (in Section 2.3).

In particular, given the penalty parameters $\boldsymbol{\eta}_k$, we want to simultaneously enforce two (sub)conditions.

Adaptive accuracy condition. We first compute a threshold

$$\delta_k^{\text{trial}} := \frac{(0.5-\beta)\eta_{2,k}}{(1+\eta_{1,k}+\eta_{2,k})\Upsilon_k^2 \Psi_k^2}, \tag{8}$$

where $\beta \in (0, 0.5)$ is an input parameter of the algorithm used in the Armijo condition in line search (cf. (13)), and

$$\Psi_k = \frac{20(\|B_k\|^2 \vee 1)}{(\xi_B \wedge 1)(\sigma_{1,k}^2 \wedge 1)}, \quad \Upsilon_k = \|G_k\| \vee \|H_k\| \vee 1, \tag{9}$$

with $\sigma_{1,k}$ being the least singular value of G_k and $\xi_B > 0$ being another input parameter used for constructing B_k (cf. Assumption 3.2). With any $0 < \delta_k \le \delta_k^{\text{trial}}$ and any prespecified sequence $\{\theta_k\} \subseteq (0, 1]$, we require $\boldsymbol{r}_{k,j}$ to satisfy

$$\|\boldsymbol{r}_{k,j}\| \le \theta_k \delta_k \|\nabla \mathcal{L}_k\| / (\|\Gamma_k\| \Psi_k). \tag{10}$$

We explain four aspects of the above condition. First, $\|\Gamma_k\|$ in (10) makes the approximation error $\|\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k\|$ relating to $\|\boldsymbol{r}_{k,j}\|$ be bounded by a factor of the exact direction $\|\Delta\boldsymbol{z}_k\|$, seen from $\|\nabla\mathcal{L}_k\| = \|\Gamma_k \Delta\boldsymbol{z}_k\| \le \|\Gamma_k\| \cdot \|\Delta\boldsymbol{z}_k\|$. Second, the factor 20 in Ψ_k is a conservative, artificial constant coming from the proof, which may be significantly reduced by finer analysis. Third, the condition $\delta_k \le \delta_k^{\text{trial}}$ is to ensure that the algorithm selects a unit stepsize locally. In fact, under standard assumptions, we show in Lemma 3.8 that δ_k^{trial} is uniformly lower bounded away from zero, so that δ_k does not have to converge to zero; instead, for k large enough, δ_k stabilizes. Fourth, θ_k is introduced to enhance the flexibility of the accuracy condition. When $\theta_k = \theta \in (0, 1]$, $\forall k$, we show the algorithm exhibits local linear convergence. When θ_k decays to zero, we actually gradually sharpen the accuracy condition (10) (as δ_k finally stabilizes); and the algorithm exhibits local superlinear convergence with a rate depending on θ_k. See Section 4 for the results of different θ_k.

Descent direction condition. We require $\tilde{\Delta}\boldsymbol{z}_{k,j}$ to be a descent direction of $\mathcal{L}_{\boldsymbol{\eta}_k}$ at $(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$. Specifically, we require

$$(\nabla \mathcal{L}_{\boldsymbol{\eta}_k}^k)^T \tilde{\Delta}\boldsymbol{z}_{k,j} \le -\eta_{2,k} \|\nabla \mathcal{L}_k\|^2 / 2. \tag{11}$$

In the above condition, the left-hand side is the reduction of the exact augmented Lagrangian, while the right-hand side regards the KKT residual. We note that both (10) and (11) involve the penalty parameters $\boldsymbol{\eta}_k$, and not every $\boldsymbol{\eta}_k$ can make (10) and (11) satisfied simultaneously. For example, (11) may not hold for some $\boldsymbol{\eta}_k$ even with exact direction $\Delta\boldsymbol{z}_k$. This illustrates the necessity for choosing suitable $\boldsymbol{\eta}_k$. We resolve this difficulty using double While loops.

Outer While loop: we check if the step acceptance condition holds, i.e., (10) and (11) hold simultaneously. Thus, after we break out the outer While loop, we always have a favorable direction $\tilde{\Delta}\boldsymbol{z}_k$ and suitable penalty parameters $\boldsymbol{\eta}_k$.

Inner While loop: with $\boldsymbol{\eta}_k$ and $\delta_k \le \delta_k^{\text{trial}}$, we repeat (6) until (10) is triggered. Then, we check if (11) holds. If (11) holds, we break the outer While loop. Otherwise, we update the parameters in an adaptive way:

$$\begin{aligned} &\eta_{1,k} \leftarrow \eta_{1,k}\nu^2, && \eta_{2,k} \leftarrow \eta_{2,k}/\nu, \\ &\text{compute } \delta_k^{\text{trial}} \text{ as (8)}, && \delta_k \leftarrow (\delta_k/\nu^4 \wedge \delta_k^{\text{trial}}), \end{aligned} \tag{12}$$

with $\nu > 1$ being any factor larger than 1, and go back to reiterating (6) from the latest inexact direction with new parameters $\boldsymbol{\eta}_k, \delta_k$, which lead to a new condition (10). The motivation of (12) is to decrease $\eta_{2,k}$ and the ratio $\delta_k \eta_{1,k}/\eta_{,k2}$, but increase the product $\eta_{1,k}\eta_{2,k}$, and have $\delta_k \le \delta_k^{\text{trial}}$ (see Lemma 3.7). Thus, we use different powers of ν for the update. See Algorithm 1 Lines 6-14 for double While loops.

Algorithm 1 AdaSketch-Newton Method

1: **Input:** initial iterate $\boldsymbol{z}_0$; sequence $\{\theta_k\} \subseteq (0,1]$; scalars $\eta_{1,0}, \eta_{2,0}, \xi_B > 0$, $\delta_0 \in (0,1)$, $\beta \in (0, 0.5)$, $\nu > 1$;
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Compute f_k, ∇f_k, c_k, G_k, H_k, and generate B_k;
4: Compute Ψ_k, Υ_k by (9) and δ_k^{trial} by (8);
5: Set $\delta_k \leftarrow \delta_k \wedge \delta_k^{\text{trial}}$, $\tilde{\Delta}\boldsymbol{z}_k \leftarrow \mathbf{0}$; compute $\boldsymbol{r}_k$ by (7);
6: **while** *Step Acceptance Condition* does not hold **do**
7: **while** $\|\boldsymbol{r}_k\| > \theta_k \delta_k \|\nabla \mathcal{L}_k\| / (\|\Gamma_k\| \Psi_k)$ **do**
8: Generate $S \sim \mathcal{P}$ and update $\tilde{\Delta}\boldsymbol{z}_k$ by (6);
9: Compute $\boldsymbol{r}_k$ by (7);
10: **end while**
11: **if** $(\nabla \mathcal{L}_{\boldsymbol{\eta}_k}^k)^T \tilde{\Delta}\boldsymbol{z}_k > -\eta_{2,k} \|\nabla \mathcal{L}_k\|^2 / 2$ **then**
12: Update $\eta_{1,k}, \eta_{2,k}, \delta_k^{\text{trial}}, \delta_k$ as (12);
13: **end if**
14: **end while**
15: Select α_k to satisfy (13); update the iterate by (14);
16: Set $\eta_{1,k+1} \leftarrow \eta_{1,k}$, $\eta_{2,k+1} \leftarrow \eta_{2,k}$, and $\delta_{k+1} \leftarrow \delta_k$;
17: **end for**

We now discuss the computational complexity of the double While loops. Once the adaptive accuracy condition (10) is triggered by the inner While loop, only checking the descent direction condition (11) is left for the outer While loop. Thus, the double While loops cost $O(\#$ of outer loop iterations $\times$ (cost of inner While loop $+ (n+m)))$. Furthermore, the cost of inner While loop (i.e., performing (6) until (10) is satisfied) is $O(\#$ of inner loop iterations $\times (n+m)^2)$ with dense sketching vectors (e.g., Gaussian) and $O(\#$ of inner loop iterations$\times(n+m))$ with sparse sketching vectors (e.g., Kaczmarz). As we will see in Lemmas 3.4, 3.5, and 3.6, the number of inner loop iterations is random. As for the outer loop iterations, there is no precise count when the algorithm is in the phase of adaptively selecting the parameters ($\boldsymbol{\eta}_k$, δ_k). However, when k is large enough, there will be only one outer loop iteration; as shown in Lemmas 3.7 and 3.8, all parameters ($\boldsymbol{\eta}_k$, δ_k) will be stabilized after large k, hence, (11) is always satisfied as long as (10) is satisfied.

2.3. Line Search and Iterate Update

We select the stepsize α_k by doing line search and enforcing the Armijo condition on the *exact augmented Lagrangian*:

$$\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{z}_k + \alpha_k \tilde{\Delta}\boldsymbol{z}_k) \leq \mathcal{L}_{\boldsymbol{\eta}_k}^k + \alpha_k \beta (\nabla \mathcal{L}_{\boldsymbol{\eta}_k}^k)^T \tilde{\Delta}\boldsymbol{z}_k, \quad (13)$$

where $\boldsymbol{z}_k = (\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$. Then, the iterate is updated as

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha_k \tilde{\Delta}\boldsymbol{z}_k. \quad (14)$$

The full design of AdaSketch-Newton is in Algorithm 1.

3. Well-posedness and Global Convergence

In this section, we first study the well-posedness of Algorithm 1 by showing that the step acceptance condition in Section 2.2 is always triggered with a finite j; thus, the double While loops in Algorithm 1 Lines 6-14 terminate in finite time. We then show a global almost sure convergence guarantee — starting from any initial point, the KKT residual $\|\nabla \mathcal{L}_k\|$ converges to zero *almost surely*. Here, the randomness plays a key role in the analysis, because the inexact direction is calculated by a randomized solver and all the algorithmic components that are affected by the direction are also random. For example, the step acceptance condition and Armijo condition are governed by random sketching matrices. We begin by stating the assumptions.

Assumption 3.1. The iterates $\{\boldsymbol{x}_k, \boldsymbol{\lambda}_k\}_{k \geq 0}$ are contained in a convex compact set $\mathcal{X} \times \Lambda$ such that over $\mathcal{X}$, the objective f and constraint c are twice continuously differentiable with Hessians being Lipschitz continuous.

Assumption 3.2. There exist absolute constants ξ_G, ξ_B, Υ_B such that (i) the Jacobian G_k has full row rank with $G_k G_k^T \succeq \xi_G I$, $\forall k$; (ii) the modified Hessian B_k satisfies $\boldsymbol{u}^T B_k \boldsymbol{u} \geq \xi_B \|\boldsymbol{u}\|^2$ for any $\boldsymbol{u} \in \{\boldsymbol{u} : G_k \boldsymbol{u} = 0\}$ and $\|B_k\| \leq \Upsilon_B$.

Assumption 3.3. There exists a constant $\pi \in (0, 1]$ such that the sketching matrices $S_{k,j} \sim S$, *iid*, satisfy $P(S^T \boldsymbol{u} \neq \mathbf{0}) \geq \pi$ for any $\boldsymbol{u} \in \mathbb{R}^{n+m} \backslash \{\mathbf{0}\}$.

All three assumptions are mild, standard, and commonly imposed in the literature. In particular, Assumptions 3.1 and 3.2 are required for the analysis of exact, inexact, stochastic SQP methods (Bertsekas, 1982; Boggs & Tolle, 1995; Nocedal & Wright, 2006; Byrd et al., 2008; 2010; Na et al., 2023; 2022a). An alternative statement of Assumption 3.1 is to assume the iterates lie in a convex open set, and the objective f, the constraint c, together with their gradients and Hessians, are Lipschitz continuous and bounded over that set (Curtis et al., 2021b). Assumption 3.2 is required to ensure that the Newton system (4) has a unique solution. Assumption 3.3 is a condition on the sketching distribution. It holds for various choices of sketching matrices. For example, in randomized Kaczmarz method, $S = \boldsymbol{e}^i \in \mathbb{R}^{n+m}$, the i-th canonical basis, with equal probability. This choice satisfies Assumption 3.3 with $\pi = 1/(n+m)$. It is also immediate to see that Assumption 3.3 holds with $\pi = 1$ for any continuous sketching distribution (e.g., Gaussian sketching). We do not impose any conditions on the sketching dimension d; thus, we can set $d = 1$ (i.e., pseudoinverse in (6) becomes reciprocal) for sake of low flops complexity.

We first present Lemmas 3.4 and 3.5, which are the building blocks for showing the well-posedness of Algorithm 1.

Lemma 3.4. *For any outer iteration k and inner iteration j, let $Q_{k,j} \in \mathbb{R}^{(n+m)\times d}$ be a matrix that has orthonormal columns spanning the space $row(S_{k,j}^T \Gamma_k)$; and let $\{j_l^k\}_{l \geq 0}$ be a subsequence of the inner iteration j, where $j_0^k = 0$ and j_l^k, $l \geq 1$, is recursively defined to satisfy*

$$col(Q_{k,j_{l-1}^k}) \cup \cdots \cup col(Q_{k,j_l^k - 1}) = \mathbb{R}^{n+m}.$$

Let L be any positive integer. Under Assumptions 3.2, 3.3, and for any k, let us suppose Algorithm 1 reaches $\boldsymbol{z}_k$. Then, the event

$$\mathcal{A}_k = \cap_{l=1}^{L}\{j_l^k < \infty\} \tag{15}$$

happens with probability one.

By Lemma 3.4, we know that the inner iteration j has a subsequence j_l^k such that the union of the space $row(S_{k,j}^T \Gamma_k)$ from $j = j_l^k$ to $j = j_{l+1}^k - 1, \forall l \geq 0$, is the full space $\mathbb{R}^{n+m}$. Such a full space expansion property is critical to show that the random approximation error, although does not monotonically decrease, has a decreasing subsequence.

Lemma 3.5 (A subsequence of error decays linearly)**.** *Under Assumptions 3.2, 3.3, let us suppose the event $\mathcal{A}_k$ in (15) happens. Then, there exists a sequence of scalars $\{\gamma_{k,l}\}_{l=1}^{L}$ such that $\gamma_{k,l} \sim \gamma_k$ is an iid realization of a random variable $\gamma_k \in [0, 1)$, and we have for $1 \leq l \leq L$,*

$$\|\tilde{\Delta}\boldsymbol{z}_{k,j_l^k} - \Delta\boldsymbol{z}_k\| \leq \gamma_{k,l}\|\tilde{\Delta}\boldsymbol{z}_{k,j_{l-1}^k} - \Delta\boldsymbol{z}_k\|.$$

Lemma 3.5 suggests that a subsequence of the approximation error, $\{\|\tilde{\Delta}\boldsymbol{z}_{k,j_l^k} - \Delta\boldsymbol{z}_k\|\}_{l\geq 0}$, decays linearly with the rate being an iid copy of some random variable $\gamma_k \in [0, 1)$. We should mention that the statement of Lemma 3.5 is deterministic (i.e., $\{\gamma_{k,l}\}_l$ are realized) since we suppose the event $\mathcal{A}_k$ happens in the statement.

Lemma 3.5 directly leads to the result that the adaptive accuracy condition (10) can be satisfied with finite inner iterations, although the iteration number may be random.

Lemma 3.6 (Well-posedness of accuracy condition)**.** *Under Assumptions 3.1, 3.2, 3.3, and for any outer iteration k and any $\delta_k, \theta_k > 0$, let us suppose Algorithm 1 reaches $\boldsymbol{z}_k$. Then, with probability one, there exists a finite number $J_k < \infty$ such that the accuracy condition (10) with $\delta_k, \theta_k > 0$ can be satisfied by iterating (6) for J_k times.*

From the proof of Lemma 3.6, we know the right-hand side of the accuracy condition (10) can be replaced by any positive upper bound, and the condition is still satisfied with (random) finite inner iterations. Lemma 3.6 suggests that the inner while loop (cf. Algorithm 1, Lines 7-10) always terminates properly. We next investigate the descent direction condition (11) to complete the well-posedness study.

Lemma 3.7 (Well-posedness of descent direction condition)**.** *Under Assumptions 3.1, 3.2, 3.3, and for any outer iteration k, we let $\tilde{\Delta}\boldsymbol{z}_{k,j}$ be the inexact solution to (4) that satisfies (10). Then, there exists a constant $\Upsilon = \Upsilon(\xi_G, \xi_B, \Upsilon_B) > 0$ large enough such that the descent direction condition (11) is satisfied as long as*

$$\eta_{1,k}\eta_{2,k} \geq \Upsilon \quad \textit{and} \quad \eta_{2,k} \vee \delta_k\eta_{1,k}/\eta_{2,k} \leq 1/\Upsilon. \tag{16}$$

Lemma 3.7 suggests that the descent direction condition (11) holds as long as $\eta_{1,k}$ is large enough and $\eta_{2,k}, \delta_k$ are small enough. By our updating rule (12), we increase the quantity $\eta_{1,k}\eta_{2,k}$ and decrease the quantities $\eta_{2,k}$ and $\delta_k\eta_{1,k}/\eta_{2,k}$ by a factor of $\nu > 1$ whenever (11) is not satisfied. Thus, (16) (and hence (11)) will be finally satisfied. Combining Lemmas 3.6 and 3.7, we have now shown the double While loops (cf. Algorithm 1, Lines 6-14) terminate properly.

We next study the behavior of adaptive penalty parameters.

Lemma 3.8 (Stability of adaptive parameters)**.** *Under Assumptions 3.1, 3.2, 3.3, with probability one, there exists an iteration threshold K such that the parameters $(\eta_{1,k}, \eta_{2,k}, \delta_k)$ are stabilized after K iterations, that is, $(\eta_{1,k}, \eta_{2,k}, \delta_k) = (\eta_{1,K}, \eta_{2,K}, \delta_K)$, $\forall k \geq K$.*

The stability of penalty parameters is crucial for global convergence. Due to our adaptive design, the algorithm chooses suitable parameters automatically. Thus, the augmented Lagrangian merit function for line search may differ from step by step. Then, we cannot accumulate the decreases across the steps (since each step may decrease a different function). Lemma 3.8 suggests that our adaptive design leads to a stabilized augmented Lagrangian in the end; thus we can accumulate all the decreases on the tail.

We then show the stepsize α_k has a uniform lower bound.

Lemma 3.9 (Armijo condition)**.** *Under Assumptions 3.1, 3.2, 3.3, with probability one, there exists $\alpha_{\min} > 0$ such that $\alpha_k \geq \alpha_{\min}$, $\forall k \geq 0$.*

With all of the above lemmas, we now establish the global almost sure convergence of Algorithm 1 in Theorem 3.10.

Theorem 3.10 (Global convergence)**.** *Under Assumptions 3.1–3.3, with probability one, $\|\nabla\mathcal{L}_k\| \to 0$ as $k \to \infty$.*

Compared to randomized inexact Newton methods for unconstrained convex optimization (Qu et al., 2016; Doikov & Richtárik, 2018; Luo et al., 2016; Gower et al., 2019; Hanzely et al., 2020), we establish the convergence of KKT residual that interprets the constraint violation. Compared to deterministic inexact Newton methods for constrained optimization (Byrd et al., 2008; 2010), our global result holds almost surely, instead of deterministically; and our next local analysis also complements the missing part in their studies.

4. Local Convergence

In this section, we establish the local convergence rate of Algorithm 1. We first show that when we set $\theta_k = \theta \in (0, 1]$, Algorithm 1 exhibits local linear convergence. We then show that when we let θ_k decay to zero, i.e., when we gradually sharpen the adaptive accuracy condition (10), Algorithm 1 exhibits local superlinear convergence.

We first present two additional assumptions that are necessary for local analysis.

Assumption 4.1. We assume f and each coordinate of c are thrice continuously differentiable over $\mathcal{X}$.

Assumption 4.2 (Hessian modification vanishes)**.** We assume $\|H_k - B_k\| = O(\tau_k)$ for a sequence $\tau_k \to 0$ as $k \to \infty$.

Assumption 4.1 strengthens Assumption 3.1 by requiring one more derivative — the third derivative — of f and c to exist. This condition is standard for local analysis when using the exact augmented Lagrangian merit function in the algorithm (Bertsekas, 1982; Zavala & Anitescu, 2014; Na et al., 2021; Na, 2021), because the Hessian of the augmented Lagrangian $\nabla^2_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}}$ requires the existence of $\nabla^3 f$ and $\nabla^3 c$. Fortunately, the third derivatives are never computed in the algorithm. Assumption 4.2 assumes the Hessian modification gradually vanishes, which is also standard in the SQP literature (Boggs & Tolle, 1995; Nocedal & Wright, 2006). It is worth mentioning that Assumption 4.2 implies that H_k satisfies Assumption 3.2-(ii) in the limit, which is known as the second-order sufficient conditions. There are multiple ways to generate B_k that satisfies Assumptions 3.2 and 4.2. One example is to test the positiveness of $Z_k^T H_k Z_k$, where the columns of $Z_k \in \mathbb{R}^{n\times(n-m)}$ span the null space of Jacobian G_k. If $Z_k^T H_k Z_k$ is positive definite, then we set $B_k = H_k$; otherwise we set $B_k = H_k + (\xi_B + \|H_k\|)I$. By this way, we have $\tau_k = 0$ for all large enough k.

Theorem 4.3 (Local linear convergence)**.** *Let $\boldsymbol{z}^\star$ be a local solution to* (1) *and $\theta_k = \theta \in (0, 1]$, $\forall k$. Under Assumptions 3.1–3.3, 4.1, 4.2 and suppose $\boldsymbol{z}_k \to \boldsymbol{z}^\star$, for all sufficiently large k, we have $\alpha_k = 1$ and (noting that $\theta\delta_K < 1$)*

$$\|\boldsymbol{z}_{k+1} - \boldsymbol{z}^\star\| \le (1+\varphi)\theta\delta_K\|\boldsymbol{z}_k - \boldsymbol{z}^\star\|, \quad \textit{for any } \varphi > 0.$$

Corollary 4.4 (Local superlinear convergence)**.** *Let $\boldsymbol{z}^\star$ be a local solution to* (1) *and θ_k be any input sequence such that $\theta_k \to 0$ as $k \to \infty$. Under Assumptions 3.1–3.3, 4.1, 4.2 and suppose $\boldsymbol{z}_k \to \boldsymbol{z}^\star$, for all sufficiently large k, we have $\alpha_k = 1$ and that*

$$\|\boldsymbol{z}_{k+1} - \boldsymbol{z}^\star\| \le O(\theta_k + \tau_k)\,\|\boldsymbol{z}_k - \boldsymbol{z}^\star\| + O\left(\|\boldsymbol{z}_k - \boldsymbol{z}^\star\|^2\right).$$

Since θ_k is a factor of accuracy condition on the sketching solver (cf. (10)), and δ_k is stabilized to δ_K, we know a decaying input sequence θ_k suggests Algorithm 1 performs more inner iterations in expense of a faster local rate.

From the global convergence in Theorem 3.10, we know that Algorithm 1 generates iterates that converge to any stationary points. In contrast, as we mentioned earlier, we assume that the second-order sufficient conditions hold at $\boldsymbol{z}^\star$ for the local convergence results. Thus, Theorem 4.3 and Corollary 4.4 indicate that the iterates generated by Algorithm 1 will exhibit linear/superlinear local rates, provided the stationary point is a second-order stationary point.

5. Experiments

We benchmark AdaSketch-Newton (Algorithm 1) on nonlinear problems in CUTEst collection set (Gould et al., 2014), on constrained logistic regression with data from LIBSVM (Chang & Lin, 2011), and on a PDE-constrained problem (Hintermüller et al., 2002). We compare the performance of Algorithm 1 with that of two inexact SQP methods designed for constrained problems with deterministic solvers: Algorithm B of Byrd et al. (2008) and its adaptive modification. The two methods are detailed in Algorithms 2 and 3 in Appendix C. Compared to Algorithm 2 that uses a fixed bound throughout all iterations, Algorithm 3 adaptively controls the accuracy of a deterministic solver and can be seen as a deterministic version of Algorithm 1. However, both methods employ an nonsmooth ℓ_1 penalized merit function that differs from (2). As another baseline for the comparison, we also consider standard augmented Lagrangian method with an inexact Newton subproblem solver (see Algorithm 4 in Appendix C). For Algorithm 1, we apply two sketching distributions: (1) Gaussian vector sketch and (2) Randomized Kaczmarz sketch, referred as AdaSketch-Newton-GV and AdaSketch-Newton-RK, respectively. As suggested in Byrd et al. (2008) and Nocedal & Wright (2006), we use GMRES (Saad & Schultz, 1986) as the deterministic solver for Algorithms 2, 3, and 4. We evaluate each algorithm with the following three criteria: (1) the KKT residual $\|\nabla\mathcal{L}_k\|$; (2) the number of objective and constraints evaluations; and (3) the number of gradient and Jacobian evaluations. Further, we assess each algorithm with the performance profile (Dolan & Moré, 2002) for CUTEst and constrained logistic regression, in which the total number of flops is used as a performance measure. For all methods, we stop iterating if

$$\|\nabla\mathcal{L}_k\| \le 10^{-4} \quad \text{OR} \quad k \ge 10^4.$$

The parameters of each algorithm are specified as follows. (We test the sensitivity to parameters for Algorithm 1 in Section 5.4).

Alg. 1: $\eta_{2,0} = \delta_0 = \xi_B = \beta = 0.1$, $\eta_{1,0} = \theta_k = 1$, $\nu = 1.5$.
Alg. 2: we follow the exact same setup as Byrd et al. (2008). In particular, with their notation, $\eta = 10^{-8}$, $\kappa_1 = \epsilon = \tau = \xi_B = 0.1$, $\pi_0 = \kappa = 1$, $\beta = 1 \vee \|\nabla\mathcal{L}_0\|_1/(\|c_0\|_1 + 1)$.
Alg. 3: we follow the same setup as Algorithm 2 and let $\eta = 10^{-8}$, $\kappa_0 = \xi_B = 0.1$, $\pi_0 = 1$, $\nu = 1.5$.
Alg. 4: $\kappa = 10^{-4}$, $\tau_0 = \eta = 0.1$, $\mu_0 = 1$, $\nu_\mu = 1.5$, $\nu_\tau = 0.5$.

Our code for the implementation is available at `https://github.com/IlgeeHong/AdaSketch-Newton`.

5.1. CUTEst

Among all the problems collected in CUTEst test set, we select the problems that have a non-constant f with $n < 1000$, and contain only equality constraints with Jacobian G_k being full rank at each step for all three algorithms. This leads to a total of 47 problems. For each problem, the initial iterate $\boldsymbol{z}_0 = (\boldsymbol{x}_0, \boldsymbol{\lambda}_0)$ is provided by CUTEst package, and we average results over 10 independent runs for Algorithm 1.

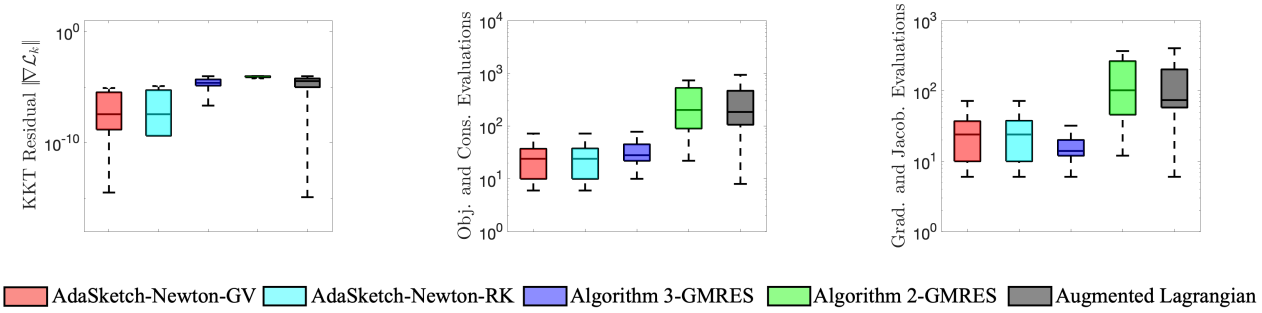

Figure 1: The boxplots of the KKT residual, the number of objective and constraints evaluations, and the number of gradient and Jacobian evaluations for AdaSketch-Newton, Algorithm 2, Algorithm 3, and Algorithm 4 on CUTEst problems.

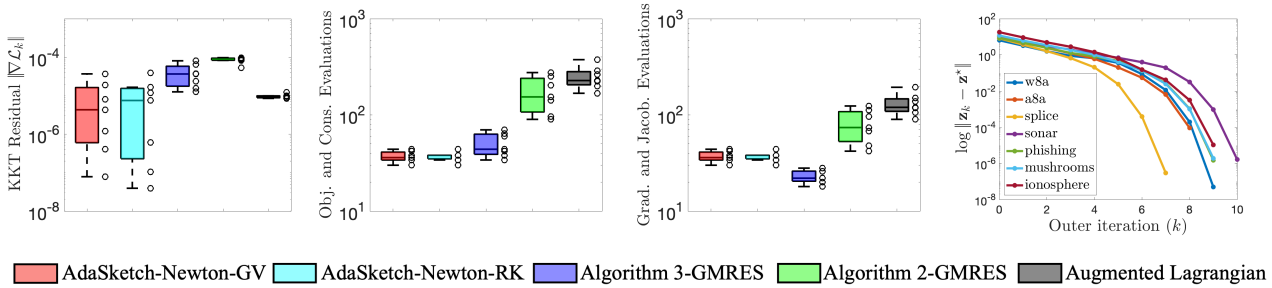

Figure 2: The boxplots of the KKT residual, the number of objective and constraints evaluations, and the number of gradient and Jacobian evaluations for AdaSketch-Newton, Algorithms 2, Algorithm 3, and Algorithm 4 on 7 LIBSVM datasets. The right figure shows the decay trajectories of the log error $\log \|\boldsymbol{z}_k - \boldsymbol{z}^\star\|$, where each line corresponds to each LIBSVM dataset ($\boldsymbol{z}^\star$ is estimated by the last iterate of the IPOPT solver (Wächter & Biegler, 2006)).

The boxplots over 47 problems of the three criteria for the five methods are shown in Figure 1. From Figure 1, we observe that AdaSketch-Newton (both GV and RK) outperforms Algorithms 2 and 3 in terms of the KKT residual and the number of objective and constraints evaluations. This observation is expected since AdaSketch-Newton employs a smooth exact augmented Lagrangian as the merit function, with an adaptive step acceptance condition to ensure a fast local convergence. Our design leads to steeper decreases on the merit function and fewer outer iterations. Compared to Algorithms 2 and 3, AdaSketch-Newton requires gradient and Jacobian evaluations to evaluate the merit function (cf. (2)), which are not needed for evaluating the ℓ_1 penalized merit function used in the other two methods. Despite this fact, we observe in Figure 1 that AdaSketch-Newton enjoys a competitive performance on the number of gradient and Jacobian evaluations with Algorithm 3 and even performs better than Algorithm 2. In addition, we also note that our adaptive modification of Algorithm 2 (i.e., Algorithm 3) is superior to Algorithm 2 in all three criteria, which illustrates the effectiveness of our adaptive technique. Finally, we see in Figure 1 that the augmented Lagrangian method (Algorithm 4) underperforms AdaSketch-Newton (both GV and RK) in terms of all three criteria.

The performance profiles over 47 problems of the total number of flops for the five methods are presented in the left-hand side of Figure 3 in Appendix C. From the figure, we see that the two AdaSketch-Newton methods are the fastest, followed by Algorithm 3. Algorithm 2 and the augmented Lagrangian method are much slower than all three adaptive methods. These results show that AdaSketch-Newton (both GV and RK) is the most effective among Algorithms 2, 3, and 4 in terms of the total number of flops.

5.2. Constrained Logistic Regression

We consider equality-constrained logistic regression problems of the form

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \log\left(1+\exp\left(-y_i \cdot \boldsymbol{d}_i^T \boldsymbol{x}\right)\right)$$
$$\text{s.t.} \;\; A\boldsymbol{x} = \boldsymbol{b}, \;\; \|\boldsymbol{x}\|^2 = 1,$$

where $(y_i, \boldsymbol{d}_i) \in \{-1, 1\} \times \mathbb{R}^n$ are the i-th data point and $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are linear constraint coefficients. We consider 7 binary classification datasets from LIBSVM for which $12 \le n \le 10^3$ and $256 \le N \le 10^5$. Details of the datasets are given in Table 1. For linear constraint, we set $m = 10$ and randomly generate each entry of A and $\boldsymbol{b}$ from a standard normal distribution. Combining with unit norm constraint, we have 11 equality constraints in total. For all datasets, we set $\boldsymbol{z}_0$ to be all-one vector and, again,

average results over 10 independent runs for Algorithm 1.

The boxplots (with points) over 7 datasets for the five methods and the decay trajectory of the log error ($\log \|\boldsymbol{z}_k - \boldsymbol{z}^\star\|$) for AdaSketch-Newton-RK are shown in Figure 2. From the figure, we have similar observations to CUTEst. AdaSketch-Newton (both GV and RK) outperforms Algorithm 3 in terms of the KKT residual and the number of objective and constraints evaluations, and it performs better than Algorithm 3 and the augmented Lagrangian method in terms of all three criteria. Algorithm 3 is superior to Algorithm 2 in all three criteria. The trajectory plots of all 7 datasets show that the log error decays at least linearly, that is consistent with our theoretical analysis in Section 4.

The performance profiles over 7 datasets of the total number of flops for the five methods are presented in the right-hand side of Figure 3 in Appendix C. From the figure, we see all three adaptive methods (two AdaSketch-Newton, Algorithm 3) are more effective than Algorithm 2 and the augmented Lagrangian in terms of the total number of flops.

Table 1: Dataset Statistics

DATASET	FEATURE DIMENSION (n)	# DATA POINTS (N)
W8A	300	49,749
A9A	123	32,561
SPLICE	60	1,000
SONAR	60	208
PHISHING	68	11,055
MUSHROOMS	112	8,124
IONOSPHERE	34	351

5.3. PDE-constrained Problem

We consider optimal control problem with Dirichlet boundary conditions of the form

$$\min_{x,y} \ \frac{1}{2}\|x-u\|_{L^2(\Omega)}^2 + \frac{\zeta}{2}\|y\|_{L^2(\Omega)}^2$$
$$\text{s.t.} \ \ -\Delta x = y \text{ in } \Omega, \ x = \mathbf{0} \text{ on } \partial\Omega,$$

where $u \in L^2(\Omega)$ is a reference function and $\zeta > 0$ is a regularization parameter. We discretize Ω on an evenly spaced $N \times N$ grid. For reference function $u = [u_{ij}]_{i,j=1}^{N,N}$, we follow Curtis et al. (2021b) and choose for all $(i,j) \in \{1,\dots,N\} \times \{1,\dots,N\}$ the following:

$$u_{ij} = \sin\left(4 + \tfrac{\epsilon_N}{\epsilon_S}\left(i - \tfrac{N+1}{2}\right)\right) + \cos\left(3 + \tfrac{\epsilon_N}{\epsilon_S}\left(j - \tfrac{N+1}{2}\right)\right),$$

where $\epsilon_N, \epsilon_S > 0$. We select $N = 3$ and $\zeta = 10^{-1}$, and as suggested in Curtis et al. (2021b), we choose $\epsilon_N = 10^{-1}$ and $\epsilon_S = \sqrt{15}$. We set $\boldsymbol{z}_0 = (x_0, y_0, \lambda_0)$ to be all-one matrix and average over 10 independent runs for Algorithm 1.

The numerical results for the five methods are summarized in Table 2 in Appendix C. From Table 2, we reconfirm our observations from CUTEst and constrained logistic regression; AdaSketch-Newton (both GV and RK) outperforms Algorithms 2, 3, and 4 in terms of all three criteria.

5.4. Sensitivity to Parameters

We test the sensitivity of AdaSketch-Newton-GV to four input parameters, $(\eta_{1,0}, \eta_{2,0}, \delta_0, \beta)$, on 47 CUTEst problems used in Section 5.1. Here, $(\eta_{1,0}, \eta_{2,0})$ are the initial penalty parameters of the augmented Lagrangian; δ_0 is the initial threshold parameter that controls the accuracy of the randomized iterative sketching solver; and β is a parameter in the Armijo condition in line search. As set in Section 5.1, the default values are $(\eta_{1,0}, \eta_{2,0}, \delta_0, \beta) = (1, 0.1, 0.1, 0.1)$. Here, we choose a larger $\eta_{1,0}$ and a smaller $\eta_{2,0}$ to make (2) close to a standard augmented Lagrangian. We vary the four parameters in ranges as follows: $\eta_{1,0} \in \{0.1, 1, 10\}$, $\eta_{2,0} \in \{0.01, 0.1, 1\}$, $\delta_0 \in \{0.01, 0.1, 0.9\}$, $\beta \in \{10^{-7}, 10^{-3}, 0.1\}$. When we change one parameter, the other three are set as default. The results are summarized in Figure 4 in Appendix C. From Figure 4, we note that, for all three criteria, there are only very marginal differences in the performance of AdaSketch-Newton for different parameter settings. Thus, we conclude that, by adaptively choosing suitable penalty parameters within the algorithm, our method enjoys a robust performance to the tuning parameters.

6. Conclusion

We proposed an adaptive inexact Newton method, called AdaSketch-Newton, for solving constrained nonlinear optimization problems. At each step, the method applies a randomized iterative sketching solver to solve the Lagrangian Newton system inexactly, and employs an exact augmented Lagrangian merit function for selecting the stepsize via line search. The method adaptively controls the accuracy of the sketching solver, and selects suitable penalty parameters of the augmented Lagrangian. Under mild assumptions, we established the global almost sure convergence guarantee with a local linear rate for the method; and we proved that the linear rate can be accelerated to a superlinear rate if we gradually sharpen the accuracy condition, which is achievable by simply specifying a decreasing input sequence. We demonstrated the superior performance and robustness to parameters of our method via experiments on benchmark nonlinear problems, constrained logistic regression, and a PDE-constrained problem. The future directions include: (i) conducting complexity analysis, (ii) applying quasi-Newton updates for Hessians, and (iii) utilizing trust-region techniques to select the inexact directions and stepsizes jointly.

Acknowledgements. This work was supported by Laboratory Directed Research and Development (LDRD) funding from Berkeley Lab, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This work was also supported by the Intelligence Advanced Research Projects Agency (IARPA) and Army Research Office (ARO) under Contract No. W911NF-20-C-0035. The research of MK is supported in part by NSF Grant ECCS-2216912.

References

Berahas, A. S., Bollapragada, R., and Nocedal, J. An investigation of newton-sketch and subsampled newton methods. *Optimization Methods and Software*, 35(4):661–680, feb 2020. doi: 10.1080/10556788.2020.1725751.

Berahas, A. S., Curtis, F. E., O'Neill, M. J., and Robinson, D. P. A stochastic sequential quadratic optimization algorithm for nonlinear equality constrained optimization with rank-deficient jacobians. *arXiv preprint arXiv:2106.13015*, 2021a.

Berahas, A. S., Curtis, F. E., Robinson, D., and Zhou, B. Sequential quadratic optimization for nonlinear equality constrained stochastic optimization. *SIAM Journal on Optimization*, 31(2):1352–1379, jan 2021b. doi: 10.1137/20m1354556.

Bertsekas, D. P. *Constrained Optimization and Lagrange Multiplier Methods*. Elsevier, 1982. doi: 10.1016/c2013-0-10366-2.

Boggs, P. T. and Tolle, J. W. Sequential quadratic programming. *Acta Numerica*, 4:1–51, jan 1995. doi: 10.1017/s0962492900002518.

Bollapragada, R., Byrd, R. H., and Nocedal, J. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, apr 2018. doi: 10.1093/imanum/dry009.

Burke, J. V., Curtis, F. E., Wang, H., and Wang, J. Inexact sequential quadratic optimization with penalty parameter updates within the QP solver. *SIAM Journal on Optimization*, 30(3):1822–1849, jan 2020. doi: 10.1137/18m1176488.

Byrd, R. H., Curtis, F. E., and Nocedal, J. An inexact SQP method for equality constrained optimization. *SIAM Journal on Optimization*, 19(1):351–369, jan 2008. doi: 10.1137/060674004.

Byrd, R. H., Curtis, F. E., and Nocedal, J. An inexact newton method for nonconvex equality constrained optimization. *Mathematical Programming*, 122(2):273–299, oct 2010. doi: 10.1007/s10107-008-0248-3.

Chang, C.-C. and Lin, C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, apr 2011. doi: 10.1145/1961189.1961199.

Chen, C., Tung, F., Vedula, N., and Mori, G. Constraint-aware deep neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 400–415, 2018.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics-informed neural networks: Where we are and what's next. *arXiv preprint arXiv:2201.05624*, 2022.

Curtis, F. E., Jiang, H., and Robinson, D. P. An adaptive augmented lagrangian method for large-scale constrained optimization. *Mathematical Programming*, 152(1-2):201–245, apr 2014a. doi: 10.1007/s10107-014-0784-y.

Curtis, F. E., Johnson, T. C., Robinson, D. P., and Wächter, A. An inexact sequential quadratic optimization algorithm for nonlinear optimization. *SIAM Journal on Optimization*, 24(3):1041–1074, jan 2014b. doi: 10.1137/130918320.

Curtis, F. E., Robinson, D. P., and Samadi, M. An inexact regularized newton framework with a worst-case iteration complexity of $ {\mathscr o}(\varepsilon^{-3/2}) $ for nonconvex optimization. *IMA Journal of Numerical Analysis*, 39(3):1296–1327, may 2018. doi: 10.1093/imanum/dry022.

Curtis, F. E., Robinson, D. P., Royer, C. W., and Wright, S. J. Trust-region newton-CG with strong second-order complexity guarantees for nonconvex optimization. *SIAM Journal on Optimization*, 31(1):518–544, jan 2021a. doi: 10.1137/19m130563x.

Curtis, F. E., Robinson, D. P., and Zhou, B. Inexact sequential quadratic optimization for minimizing a stochastic objective function subject to deterministic nonlinear equality constraints. *arXiv preprint arXiv:2107.03512*, 2021b.

Dereziński, M. and Rebrova, E. Sharp analysis of sketch-and-project methods via a connection to randomized singular value decomposition. *arXiv preprint arXiv:2208.09585*, 2022.

Derezinski, M., Bartan, B., Pilanci, M., and Mahoney, M. W. Debiasing distributed second order optimization with surrogate sketching and scaled regularization. *Advances in Neural Information Processing Systems*, 33:6684–6695, 2020a.

Derezinski, M., Liang, F. T., Liao, Z., and Mahoney, M. W. Precise expressions for random projections: Low-rank approximation and randomized newton. *Advances in Neural Information Processing Systems*, 33:18272–18283, 2020b.

Derezinski, M., Lacotte, J., Pilanci, M., and Mahoney, M. W. Newton-less: Sparsification without trade-offs for the sketched newton update. *Advances in Neural Information Processing Systems*, 34:2835–2847, 2021.

Doikov, N. and Richtárik, P. Randomized block cubic newton method. In *International Conference on Machine Learning*, pp. 1290–1298. PMLR, 2018.

Dolan, E. D. and Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, jan 2002. doi: 10.1007/s101070100263.

Fang, Y., Na, S., Mahoney, M. W., and Kolar, M. Fully stochastic trust-region sequential quadratic programming for equality-constrained optimization problems. *arXiv preprint arXiv:2211.15943*, 2022.

Fong, D. C.-L. and Saunders, M. CG versus MINRES: An empirical comparison. *Sultan Qaboos University Journal for Science [SQUJS]*, 16(1):44, apr 2012. doi: 10.24200/squjs.vol17iss1pp44-62.

Fuji, T., Poirion, P.-L., and Takeda, A. Randomized subspace regularized newton method for unconstrained nonconvex optimization. *arXiv preprint arXiv:2209.04170*, 2022.

Gould, N. I. M., Orban, D., and Toint, P. L. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, aug 2014. doi: 10.1007/s10589-014-9687-3.

Gower, R., Kovalev, D., Lieder, F., and Richtárik, P. Rsn: Randomized subspace newton. *Advances in Neural Information Processing Systems*, 32, 2019.

Gower, R. M. and Richtárik, P. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, jan 2015. doi: 10.1137/15m1025487.

Gower, R. M., Molitor, D., Moorman, J., and Needell, D. On adaptive sketch-and-project for solving linear systems. *SIAM Journal on Matrix Analysis and Applications*, 42 (2):954–989, jan 2021. doi: 10.1137/19m1285846.

Gu, C., Zhu, D., and Pei, Y. A new inexact SQP algorithm for nonlinear systems of mixed equalities and inequalities. *Numerical Algorithms*, 78(4):1233–1253, sep 2017. doi: 10.1007/s11075-017-0421-y.

Hanzely, F., Doikov, N., Nesterov, Y., and Richtarik, P. Stochastic subspace cubic newton method. In *International Conference on Machine Learning*, pp. 4027–4038. PMLR, 2020.

Hintermüller, M., Ito, K., and Kunisch, K. The primal-dual active set strategy as a semismooth newton method. *SIAM Journal on Optimization*, 13(3):865–888, jan 2002. doi: 10.1137/s1052623401383558.

Kouri, D. P., Heinkenschloss, M., Ridzal, D., and van Bloemen Waanders, B. G. Inexact objective function evaluations in a trust-region algorithm for PDE-constrained optimization under uncertainty. *SIAM Journal on Scientific Computing*, 36(6):A3011–A3029, jan 2014. doi: 10.1137/140955665.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Lacotte, J., Wang, Y., and Pilanci, M. Adaptive newton sketch: Linear-time optimization with quadratic convergence and effective hessian dimensionality. In *International Conference on Machine Learning*, pp. 5926–5936. PMLR, 2021.

Lewis, F. L., Vrabie, D. L., and Syrmos, V. L. *Optimal Control*. John Wiley & Sons, Inc., jan 2012. doi: 10.1002/9781118122631.

Liu, Y. and Roosta, F. A newton-mr algorithm with complexity guarantees for nonconvex smooth unconstrained optimization. *arXiv preprint arXiv:2208.07095*, 2022.

Luo, H., Agarwal, A., Cesa-Bianchi, N., and Langford, J. Efficient second order online learning by sketching. *Advances in Neural Information Processing Systems*, 29, 2016.

Murray, R., Demmel, J., Mahoney, M. W., Erichson, N. B., Melnichenko, M., Malik, O. A., Grigori, L., Luszczek, P., Dereziński, M., Lopes, M. E., et al. Randomized numerical linear algebra: A perspective on the field with an eye to software. *arXiv preprint arXiv:2302.11474*, 2023.

Mutny, M., Derezinski, M., and Krause, A. Convergence analysis of block coordinate algorithms with determinantal sampling. In *International Conference on Artificial Intelligence and Statistics*, pp. 3110–3120. PMLR, 2020.

Na, S. Global convergence of online optimization for nonlinear model predictive control. *Advances in Neural Information Processing Systems*, 34:12441–12453, 2021.

Na, S. and Mahoney, M. W. Asymptotic convergence rate and statistical inference for stochastic sequential quadratic programming. *arXiv preprint arXiv:2205.13687*, 2022.

Na, S., Anitescu, M., and Kolar, M. A fast temporal decomposition procedure for long-horizon nonlinear dynamic programming. *arXiv preprint arXiv:2107.11560*, 2021.

Na, S., Anitescu, M., and Kolar, M. An adaptive stochastic sequential quadratic programming with differentiable exact augmented lagrangians. *Mathematical Programming*, pp. 1–71, jun 2022a. doi: 10.1007/s10107-022-01846-z.

Na, S., Dereziński, M., and Mahoney, M. W. Hessian averaging in stochastic newton methods achieves superlinear convergence. *Mathematical Programming*, dec 2022b. doi: 10.1007/s10107-022-01913-5.

Na, S., Anitescu, M., and Kolar, M. Inequality constrained stochastic nonlinear optimization via active-set sequential quadratic programming. *Mathematical Programming*, mar 2023. doi: 10.1007/s10107-023-01935-7.

Nocedal, J. and Wright, S. J. *Numerical Optimization (2nd Edition)*. Springer-Verlag, 2006. doi: 10.1007/b98874.

Patel, V., Jahangoshahi, M., and Maldonado, D. A. An implicit representation and iterative solution of randomly sketched linear systems. *SIAM Journal on Matrix Analysis and Applications*, 42(2):800–831, jan 2021. doi: 10.1137/19m1259481.

Pilanci, M. and Wainwright, M. J. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, jan 2017. doi: 10.1137/15m1021106.

Qu, Z., Richtárik, P., Takác, M., and Fercoq, O. Sdna: stochastic dual newton ascent for empirical risk minimization. In *International Conference on Machine Learning*, pp. 1823–1832. PMLR, 2016.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144 (1-2):1–38, dec 2014. doi: 10.1007/s10107-012-0614-z.

Roosta, F., Liu, Y., Xu, P., and Mahoney, M. W. Newton-MR: Inexact newton method with minimum residual sub-problem solver. *EURO Journal on Computational Optimization*, 10:100035, 2022. doi: 10.1016/j.ejco.2022.100035.

Roosta-Khorasani, F. and Mahoney, M. W. Sub-sampled newton methods. *Mathematical Programming*, 174(1-2): 293–326, nov 2018. doi: 10.1007/s10107-018-1346-5.

Royer, C. W. and Wright, S. J. Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2): 1448–1477, jan 2018. doi: 10.1137/17m1134329.

Royer, C. W., O'Neill, M., and Wright, S. J. A newton-CG algorithm with complexity guarantees for smooth unconstrained optimization. *Mathematical Programming*, 180(1-2):451–488, jan 2019. doi: 10.1007/s10107-019-01362-7.

Saad, Y. and Schultz, M. H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, jul 1986. doi: 10.1137/0907058.

Scott, J. and Tůma, M. Solving large linear least squares problems with linear equality constraints. *BIT Numerical Mathematics*, 62(4):1765–1787, jul 2022. doi: 10.1007/s10543-022-00930-2.

Strohmer, T. and Vershynin, R. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, apr 2008. doi: 10.1007/s00041-008-9030-4.

Wächter, A. and Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, apr 2006. doi: 10.1007/s10107-004-0559-y.

Yao, Z., Xu, P., Roosta, F., and Mahoney, M. W. Inexact nonconvex newton-type methods. *INFORMS Journal on Optimization*, 3(2):154–182, jan 2021. doi: 10.1287/ijoo.2019.0043.

Yao, Z., Xu, P., Roosta, F., Wright, S. J., and Mahoney, M. W. Inexact newton-CG algorithms with complexity guarantees. *IMA Journal of Numerical Analysis*, aug 2022. doi: 10.1093/imanum/drac043.

Yuan, R., Lazaric, A., and Gower, R. M. Sketched newton–raphson. *SIAM Journal on Optimization*, 32(3):1555–1583, jul 2022. doi: 10.1137/21m139788x.

Zavala, V. M. and Anitescu, M. Scalable nonlinear programming via exact differentiable penalty functions and trust-region newton methods. *SIAM Journal on Optimization*, 24(1):528–558, jan 2014. doi: 10.1137/120888181.

Appendix: Constrained Optimization via Exact Augmented Lagrangian and Randomized Iterative Sketching

A. Proofs of Section 3

A.1. Proof of Lemma 3.4

Throughout the proof, we fix $k \geq 0$ and suppose the algorithm reaches $\boldsymbol{z}_k = (\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$. We note that

$$\tilde{\Delta}\boldsymbol{z}_{k,j+1} \overset{(6)}{=} \tilde{\Delta}\boldsymbol{z}_{k,j} - \Gamma_k S_{k,j}(S_{k,j}^T \Gamma_k^2 S_{k,j})^\dagger S_{k,j}^T \boldsymbol{r}_{k,j} \overset{(4),(7)}{=} \tilde{\Delta}\boldsymbol{z}_{k,j} - W_{k,j}(\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k), \tag{17}$$

where $W_{k,j} = \Gamma_k S_{k,j}(S_{k,j}^T \Gamma_k^2 S_{k,j})^\dagger S_{k,j}^T \Gamma_k$. Since $W_{k,j}$ is an orthogonal projection onto $row(S_{k,j}^T \Gamma_k)$, we rewrite (17) as

$$\tilde{\Delta}\boldsymbol{z}_{k,j+1} - \Delta\boldsymbol{z}_k = \tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k - Q_{k,j}Q_{k,j}^T(\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k). \tag{18}$$

By Assumption 3.2, Γ_k is invertible (see Nocedal & Wright (2006, Lemma 16.1)). Thus, Assumption 3.3 implies that $P(S^T\Gamma_k\boldsymbol{z} \neq \boldsymbol{0}) \geq \pi$ for any $\boldsymbol{z} \in \mathbb{R}^{n+m}\backslash\{\boldsymbol{0}\}$. Given the relationship between $row(S_{k,j}^T\Gamma_k)$ and $Q_{k,j}$, we further have $P(Q_{k,j}^T\boldsymbol{z} \neq \boldsymbol{0}) = P(S_{k,j}^T\Gamma_k\boldsymbol{z} \neq \boldsymbol{0}) \geq \pi$ for any $\boldsymbol{z} \in \mathbb{R}^{n+m}\backslash\{\boldsymbol{0}\}$. Since $\{Q_{k,j}\}_j$ are independent and identically distributed, conditional on $\{j_{l-1}^k < \infty\}$, the probability that $\dim(\cup_{i=0}^{t+1} col(Q_{k,j_{l-1}^k+i}))$ grows relative to $\dim(\cup_{i=0}^{t} col(Q_{k,j_{l-1}^k+i}))$ when $\dim(\cup_{i=0}^{t} col(Q_{k,j_{l-1}^k+i})) < n+m$, is at least π. As a result, the probability that the event $\{\dim(\cup_{i=0}^{t+1} col(Q_{k,j_{l-1}^k+i})) > \dim(\cup_{i=0}^{t} col(Q_{k,j_{l-1}^k+i}))\}$ happens $n+m$ times in N iterations with $N \geq n+m$ is dominated by a negative binomial distribution. In particular, for $N \geq n+m$,

$$P(j_l^k - 1 = N + j_{l-1}^k | j_{l-1}^k < \infty) \leq \binom{N-1}{n+m-1}(1-\pi)^{N-n-m}\pi^{n+m}.$$

Taking $N \to \infty$, we have for any $l \geq 1$,

$$P(j_l^k = \infty | j_{l-1}^k < \infty) = 0.$$

Thus, for any $l \in \mathbb{N}$, $P(j_l^k < \infty | j_{l-1}^k < \infty) = 1$. Furthermore, we have

$$\begin{aligned} P\left(\cap_{l=1}^L \{j_l^k < \infty\}\right) &= P(j_1^k < \infty) \times P(j_2^k < \infty | j_1^k < \infty) \times \cdots \times P(j_L^k < \infty | j_{L-1}^k < \infty, \ldots, j_1^k < \infty) \\ &= P(j_1^k < \infty) \times P(j_2^k < \infty | j_1^k < \infty) \times \cdots \times P(j_L^k < \infty | j_{L-1}^k < \infty) = 1. \end{aligned}$$

This completes the proof.

A.2. Proof of Lemma 3.5

Let us denote $\boldsymbol{q}_{k,j,h}$ to be the h-th column of $Q_{k,j}$ for $1 \leq h \leq d$. We know for any $1 \leq l \leq L$,

$$\tilde{\Delta}\boldsymbol{z}_{k,j_l^k} - \Delta\boldsymbol{z}_k \overset{(18)}{=} \left(\Pi_{j=j_{l-1}^k}^{j_l^k-1} \Pi_{h=1}^d \left(I - \boldsymbol{q}_{k,j,h}\boldsymbol{q}_{k,j,h}^T\right)\right)(\tilde{\Delta}\boldsymbol{z}_{k,j_{l-1}^k} - \Delta\boldsymbol{z}_k).$$

Taking ℓ_2 norm on both sides, we obtain

$$\|\tilde{\Delta}\boldsymbol{z}_{k,j_l^k} - \Delta\boldsymbol{z}_k\| \leq \|\Pi_{j=j_{l-1}^k}^{j_l^k-1} \Pi_{h=1}^d \left(I - \boldsymbol{q}_{k,j,h}\boldsymbol{q}_{k,j,h}^T\right)\| \cdot \|\tilde{\Delta}\boldsymbol{z}_{k,j_{l-1}^k} - \Delta\boldsymbol{z}_k\|.$$

Let $\mathcal{F}_{k,l}$ denote the set of all matrices $F_{k,l}$, whose columns $\{f_{k,l,1}, \ldots, f_{k,l,n+m}\}$ form a maximal linearly independent subset of $\{\boldsymbol{q}_{k,j_{l-1}^k,1}, \ldots, \boldsymbol{q}_{k,j_l^k-1,d}\}$. Patel et al. (2021, Theorem 4.1) implies that

$$\|\Pi_{j=j_{l-1}^k}^{j_l^k-1} \Pi_{h=1}^d \left(I - \boldsymbol{q}_{k,j,h}\boldsymbol{q}_{k,j,h}^T\right)\| \leq \sqrt{1 - \min_{F_{k,l} \in \mathcal{F}_{k,l}} \det(F_{k,l}^T F_{k,l})} =: \gamma_{k,l}.$$

Thus, we have $\|\tilde{\Delta}\boldsymbol{z}_{k,j_l^k} - \Delta\boldsymbol{z}_k\| \leq \gamma_{k,l}\|\tilde{\Delta}\boldsymbol{z}_{k,j_{l-1}^k} - \Delta\boldsymbol{z}_k\|$. By Hadamard's inequality and the fact that $F_{k,l}^T F_{k,l}$ is positive definite, we know $0 \leq \gamma_{k,l} < 1$ (note that $\mathcal{F}_{k,l}$ is a finite set). Since $S_{k,j} \sim S$, iid, we know the distribution of $\gamma_{k,l}$ is independent of l. In particular, we let $\mathcal{Q}_{k,l} = \{\boldsymbol{q}_{k,j_{l-1}^k,1}, \ldots, \boldsymbol{q}_{k,j_l^k-1,d}\}$ and have $\mathcal{Q}_{k,1}, \ldots, \mathcal{Q}_{k,L} \sim \mathcal{Q}_k$, iid. This implies $\gamma_{k,1}, \ldots, \gamma_{k,L} \sim \gamma_k$, iid, and completes the proof.

A.3. Proof of Lemma 3.6

Throughout the proof, we fix $k \geq 0$ and suppose that the algorithm reaches $\boldsymbol{z}_k = (\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$. From Lemma 3.5, we know $P(\gamma_k = 1) = 0$; thus, there exists $\tau_k \in (0,1)$ such that $P(\gamma_k \leq \tau_k) > 0$. Let us define $\pi_k := P(\gamma_k \leq \tau_k)$. By Assumptions 3.1 and 3.2, we know $\|\Gamma_k\| \leq \Upsilon_\Gamma$ and $\Psi_k \leq \Psi$ for some Υ_Γ and $\Psi > 0$. Thus, $\theta_k \delta_k / (\|\Gamma_k\|^2 \Psi_k^2) > 0$. Let $\bar{N}$ be the smallest positive integer such that $\bar{N} \geq \log\{\theta_k \delta_k / (\|\Gamma_k\|^2 \Psi_k^2)\} / \log(\tau_k)$. Then, we have $\tau_k^{\bar{N}} \leq \theta_k \delta_k / (\|\Gamma_k\|^2 \Psi_k^2)$. We now consider a procedure where for each iteration l, we generate $\gamma_{k,l} \sim \gamma_k$ independently. Let L_k be the iteration such that

$$I\{\gamma_{k,1} \leq \tau_k\} + \cdots + I\{\gamma_{k,L_k} \leq \tau_k\} = \bar{N}.$$

We note the probability that the event $\{\gamma_{k,l} \leq \tau_k\}$ happens $\bar{N}$ times in N iterations with $N \geq \bar{N}$ is dominated by a negative binomial distribution. In particular, for $N \geq \bar{N}$,

$$P(L_k = N) \leq \binom{N-1}{\bar{N}-1} (1-\pi_k)^{N-\bar{N}} \pi_k^{\bar{N}}.$$

Taking $N \to \infty$, we have $P(L_k = \infty) = 0$. Thus, L_k is finite with probability one. We now apply Lemma 3.5. We have

$$\|\tilde{\Delta}\boldsymbol{z}_{k,j^k_{L_k}} - \Delta\boldsymbol{z}_k\| \leq \left(\Pi_{l=1}^{L_k} \gamma_{k,l}\right) \|\tilde{\Delta}\boldsymbol{z}_{k,0} - \Delta\boldsymbol{z}_k\| = \left(\Pi_{l=1}^{L_k} \gamma_{k,l}\right) \|\Delta\boldsymbol{z}_k\|. \tag{19}$$

By Assumption 3.2 and the fact that $\|(G_k G_k^T)^{-1}\| \leq 1/\sigma_{1,k}^2$ with $\sigma_{1,k}$ being the least singular value of G_k, we apply Na et al. (2021, Lemma 5.1) and have

$$\left\|\Gamma_k^{-1}\right\| \leq \frac{1}{\xi_B} + \frac{2}{\sigma_{1,k}}\left(1 + \frac{\|B_k\|}{\xi_B}\right) + \frac{1}{\sigma_{1,k}^2}\left(\|B_k\| + \frac{\|B_k\|^2}{\xi_B}\right) \leq \frac{7(\|B_k\|^2 \vee 1)}{(\xi_B \wedge 1)(\sigma_{1,k}^2 \wedge 1)} \overset{(9)}{\leq} \Psi_k. \tag{20}$$

Thus, we get

$$\|\boldsymbol{r}_{k,j^k_{L_k}}\| \overset{(7)}{=} \|\Gamma_k \tilde{\Delta}\boldsymbol{z}_{k,j^k_{L_k}} + \nabla\mathcal{L}_k\| \overset{(4)}{\leq} \|\Gamma_k\| \|\tilde{\Delta}\boldsymbol{z}_{k,j^k_{L_k}} - \tilde{\Delta}\boldsymbol{z}_k\| \overset{(19)}{\leq} \left(\Pi_{l=1}^{L_k}\gamma_{k,l}\right) \|\Gamma_k\| \|\Delta\boldsymbol{z}_k\| \overset{(4),(20)}{\leq} \left(\Pi_{l=1}^{L_k}\gamma_{k,l}\right) \|\Gamma_k\| \Psi_k \|\nabla\mathcal{L}_k\|.$$

Furthermore, we have

$$\left\{I\{\gamma_{k,1} \leq \tau_k\} + \cdots + I\{\gamma_{k,L_k} \leq \tau_k\} = \bar{N}\right\} \Rightarrow \left\{\Pi_{l=1}^{L_k}\gamma_{k,l} \leq \theta_k\delta_k / (\|\Gamma_k\|^2 \Psi_k^2)\right\} \Rightarrow \left\{\|\boldsymbol{r}_{k,j^k_{L_k}}\| \leq \theta_k\delta_k \|\nabla\mathcal{L}_k\| / (\|\Gamma_k\| \Psi_k)\right\}.$$

Finally, we let $J_k = j^k_{L_k}$ and complete the proof.

A.4. Proof of Lemma 3.7

Throughout the proof, we fix $k \geq 0$ and suppose that the algorithm reaches $\boldsymbol{z}_k = (\boldsymbol{x}_k, \boldsymbol{\lambda}_k)$. We first observe that

$$\begin{aligned}
\|\boldsymbol{r}_{k,j}\| \leq \theta_k\delta_k \|\nabla\mathcal{L}_k\| / (\|\Gamma_k\| \Psi_k) &\overset{(4)}{\overset{(7)}{\Longrightarrow}} \Psi_k \|\Gamma_k(\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k)\| \leq \theta_k\delta_k \|\nabla\mathcal{L}_k\| / \|\Gamma_k\| \\
&\overset{(20)}{\Longrightarrow} \|\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k\| \leq \theta_k\delta_k \|\nabla\mathcal{L}_k\| / \|\Gamma_k\| \\
&\overset{(4)}{\Longrightarrow} \|\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k\| \leq \theta_k\delta_k \|\Gamma_k\| \|\Delta\boldsymbol{z}_k\| / \|\Gamma_k\| \\
&\Longrightarrow \|\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k\| \leq \theta_k\delta_k \|\Delta\boldsymbol{z}_k\| \qquad (21) \\
&\Longrightarrow \|\tilde{\Delta}\boldsymbol{z}_{k,j} - \Delta\boldsymbol{z}_k\| \leq \delta_k \|\Delta\boldsymbol{z}_k\|. \qquad (22)
\end{aligned}$$

By Assumption 3.1, we know there exist constants $\Upsilon_H, \Upsilon_G > 0$ such that $\|H_k\| \leq \Upsilon_H$ and $\|G_k\| \leq \Upsilon_G$. We let $\Upsilon > 0$ be a constant such that $\Upsilon_H \vee \Upsilon_G \vee \Upsilon_B \leq \Upsilon$. We now divide $(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T \tilde{\Delta}\boldsymbol{z}_{k,j}$ into two terms as follows:

$$(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T \tilde{\Delta}\boldsymbol{z}_{k,j} = \begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k} \end{pmatrix}^T \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} \end{pmatrix} = \begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k} \end{pmatrix}^T \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} + \begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k} \end{pmatrix}^T \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} - \Delta\boldsymbol{x}_k \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} - \Delta\boldsymbol{\lambda}_k \end{pmatrix}. \tag{23}$$

For the first term, we have

$$
\begin{aligned}
\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix} &\overset{(2)}{=} \begin{pmatrix}(I+\eta_{2,k}H_k)\nabla_{\boldsymbol{x}}\mathcal{L}_k+\eta_{1,k}G_k^Tc_k\\ c_k+\eta_{2,k}G_k\nabla_{\boldsymbol{x}}\mathcal{L}_k\end{pmatrix}^T\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&= \begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}^T\begin{pmatrix}I+\eta_{2,k}H_k & \eta_{1,k}G_k^T\\ \eta_{2,k}G_k & I\end{pmatrix}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_k\\ c_k\end{pmatrix}\\
&\overset{(4)}{=} -\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}^T\begin{pmatrix}I+\eta_{2,k}H_k & \eta_{1,k}G_k^T\\ \eta_{2,k}G_k & I\end{pmatrix}\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&= -\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}^T\begin{pmatrix}(I+\eta_{2,k}H_k)B_k+\eta_{1,k}G_k^TG_k & (I+\eta_{2,k}H_k)G_k^T\\ G_k(I+\eta_{2,k}B_k) & \eta_{2,k}G_kG_k^T\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&= -\Delta\boldsymbol{x}_k{}^T\left((I+\eta_{2,k}H_k)B_k+\frac{\eta_{1,k}}{2}G_k^TG_k\right)\Delta\boldsymbol{x}_k-\frac{\eta_{1,k}}{2}\Delta\boldsymbol{x}_k{}^TG_k^TG_k\Delta\boldsymbol{x}_k\\
&\quad -\eta_{2,k}\Delta\boldsymbol{\lambda}_k{}^TG_kG_k^T\Delta\boldsymbol{\lambda}_k-\Delta\boldsymbol{\lambda}_k{}^TG_k\left(2I+\eta_{2,k}(B_k+H_k)\right)\Delta\boldsymbol{x}_k\\
&= -\Delta\boldsymbol{x}_k{}^T\left((I+\eta_{2,k}H_k)B_k+\frac{\eta_{1,k}}{2}G_k^TG_k\right)\Delta\boldsymbol{x}_k-\frac{\eta_{1,k}}{2}\left\|G_k\Delta\boldsymbol{x}_k\right\|^2-\eta_{2,k}\left\|G_k^T\Delta\boldsymbol{\lambda}_k\right\|^2\\
&\quad -\Delta\boldsymbol{\lambda}_k{}^TG_k\left(2I+\eta_{2,k}(B_k+H_k)\right)\Delta\boldsymbol{x}_k.
\end{aligned}
$$

Using (4), we have $G_k\Delta\boldsymbol{x}=-c_k$ and $G_k^T\Delta\boldsymbol{\lambda}_k=-\left(B_k\Delta\boldsymbol{x}_k+\nabla_{\boldsymbol{x}}\mathcal{L}_k\right)$. Combining these equations with the above display, we have

$$
\begin{aligned}
&\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&\quad= -\Delta\boldsymbol{x}_k{}^T\left((I+\eta_{2,k}H_k)B_k+\frac{\eta_{1,k}}{2}G_k^TG_k\right)\Delta\boldsymbol{x}_k-\frac{\eta_{1,k}}{2}\left\|c_k\right\|^2-\Delta\boldsymbol{\lambda}_k{}^TG_k\left(2I+\eta_{2,k}(B_k+H_k)\right)\Delta\boldsymbol{x}_k\\
&\qquad -\eta_{2,k}\left\|B_k\Delta\boldsymbol{x}_k+\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2\\
&\quad= -\Delta\boldsymbol{x}_k{}^T\left((I+\eta_{2,k}H_k)B_k+\frac{\eta_{1,k}}{2}G_k^TG_k\right)\Delta\boldsymbol{x}_k-\frac{\eta_{1,k}}{2}\left\|c_k\right\|^2-\Delta\boldsymbol{\lambda}_k{}^TG_k\left(2I+\eta_{2,k}(B_k+H_k)\right)\Delta\boldsymbol{x}_k\\
&\qquad -\frac{\eta_{2,k}}{2}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2+\frac{\eta_{2,k}}{2}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2-\eta_{2,k}\left\|B_k\Delta\boldsymbol{x}_k+\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2.
\end{aligned}
$$

For the last two terms in the above display, we have

$$
\begin{aligned}
&\frac{\eta_{2,k}}{2}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2-\eta_{2,k}\left\|B_k\Delta\boldsymbol{x}_k+\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2\\
&\quad= -\eta_{2,k}\left\|B_k\Delta\boldsymbol{x}_k\right\|^2-2\eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_k\nabla_{\boldsymbol{x}}\mathcal{L}_k-\frac{\eta_{2,k}}{2}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2\\
&\quad\overset{(4)}{=} -\eta_{2,k}\left\|B_k\Delta\boldsymbol{x}_k\right\|^2+2\eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_k\left(B_k\Delta\boldsymbol{x}_k+G_k^T\Delta\boldsymbol{\lambda}_k\right)-\frac{\eta_{2,k}}{2}\left\|B_k\Delta\boldsymbol{x}_k+G_k^T\Delta\boldsymbol{\lambda}_k\right\|^2\\
&\quad= \eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_k^2\Delta\boldsymbol{x}_k+\eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_kG_k^T\Delta\boldsymbol{\lambda}_k-\frac{\eta_{2,k}}{2}\left\|B_k\Delta\boldsymbol{x}_k\right\|^2-\frac{\eta_{2,k}}{2}\left\|G_k^T\Delta\boldsymbol{\lambda}_k\right\|^2\\
&\quad\le \eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_k^2\Delta\boldsymbol{x}_k+\eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_kG_k^T\Delta\boldsymbol{\lambda}_k-\frac{\eta_{2,k}}{2}\left\|G_k^T\Delta\boldsymbol{\lambda}_k\right\|^2.
\end{aligned}
$$

Combining the above two displays, we get

$$
\begin{aligned}
&\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&\quad\le -\Delta\boldsymbol{x}_k{}^T\left((I+\eta_{2,k}H_k)B_k+\frac{\eta_{1,k}}{2}G_k^TG_k\right)\Delta\boldsymbol{x}_k-\frac{\eta_{1,k}}{2}\left\|c_k\right\|^2-\Delta\boldsymbol{\lambda}_k{}^TG_k\left(2I+\eta_{2,k}(B_k+H_k)\right)\Delta\boldsymbol{x}_k\\
&\qquad -\frac{\eta_{2,k}}{2}\left\|\nabla_{\boldsymbol{x}}\mathcal{L}_k\right\|^2+\eta_{2,k}\Delta\boldsymbol{x}_k{}^TB_k^2\Delta\boldsymbol{x}_k+\eta_{2,k}\Delta\boldsymbol{\lambda}_k{}^TG_kB_k\Delta\boldsymbol{x}_k-\frac{\eta_{2,k}}{2}\|G_k^T\Delta\boldsymbol{\lambda}_k\|_2^2.
\end{aligned}
$$

Assuming $\eta_{1,k} \geq \eta_{2,k}$ at the moment and using Cauchy-Schwarz inequality, we get

$$\begin{aligned}
\begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} &\leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \Delta \boldsymbol{x}_k{}^T \left((I + \eta_{2,k}(H_k - B_k))B_k + \frac{\eta_{1,k}}{2} G_k^T G_k \right) \Delta \boldsymbol{x}_k - \frac{\eta_{2,k}}{2} \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\|^2 \\
&\quad - \Delta \boldsymbol{\lambda}_k{}^T G_k \left(2I + \eta_{2,k} H_k \right) \Delta \boldsymbol{x}_k \\
&= -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \eta_{2,k} \Delta \boldsymbol{x}_k{}^T (H_k - B_k) B_k \Delta \boldsymbol{x}_k - \Delta \boldsymbol{x}_k{}^T B_k \Delta \boldsymbol{x}_k - \frac{\eta_{1,k}}{2} \Delta \boldsymbol{x}_k{}^T G_k^T G_k \Delta \boldsymbol{x}_k \\
&\quad - \frac{\eta_{2,k}}{2} \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\|^2 - 2 \Delta \boldsymbol{\lambda}_k{}^T G_k \Delta \boldsymbol{x}_k - \eta_{2,k} \Delta \boldsymbol{\lambda}_k{}^T G_k H_k \Delta \boldsymbol{x}_k \\
&\leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 + \eta_{2,k} \|(H_k - B_k) \Delta \boldsymbol{x}_k\| \|B_k \Delta \boldsymbol{x}_k\| - \Delta \boldsymbol{x}_k{}^T B_k \Delta \boldsymbol{x}_k - \frac{\eta_{1,k}}{2} \Delta \boldsymbol{x}_k{}^T G_k^T G_k \Delta \boldsymbol{x}_k \\
&\quad - \frac{\eta_{2,k}}{2} \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\|^2 + 2 \|\Delta \boldsymbol{\lambda}_k\| \|G_k \Delta \boldsymbol{x}_k\| + \eta_{2,k} \Upsilon \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\| \|\Delta \boldsymbol{x}_k\| \\
&\leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 + 2\eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \Delta \boldsymbol{x}_k{}^T \left(B_k + \frac{\eta_{1,k}}{2} G_k^T G_k \right) \Delta \boldsymbol{x}_k - \frac{\eta_{2,k}}{2} \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\|^2 \\
&\quad + 2 \|\Delta \boldsymbol{\lambda}_k\| \|G_k \Delta \boldsymbol{x}_k\| + \eta_{2,k} \Upsilon \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\| \|\Delta \boldsymbol{x}_k\| .
\end{aligned}$$

Applying Young's inequalities for the last two terms, we obtain

$$\begin{aligned}
2 \|\Delta \boldsymbol{\lambda}_k\| \|G_k \Delta \boldsymbol{x}_k\| &\leq \frac{\eta_{2,k} \xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 + \frac{8}{\eta_{2,k} \xi_G} \|G_k \Delta \boldsymbol{x}_k\|^2 , \\
\eta_{2,k} \Upsilon \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\| \|\Delta \boldsymbol{x}_k\| &\leq \frac{\eta_{2,k}}{4} \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\|^2 + \eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 .
\end{aligned}$$

Combining the above two displays and using Assumption 3.2, we get

$$\begin{aligned}
&\begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \\
&\leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 + 3\eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \frac{\eta_{2,k}}{4} \left\| G_k^T \Delta \boldsymbol{\lambda}_k \right\|^2 + \frac{\eta_{2,k} \xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 + \frac{8}{\eta_{2,k} \xi_G} \|G_k \Delta \boldsymbol{x}_k\|^2 \\
&\quad - \Delta \boldsymbol{x}_k{}^T \left(B_k + \frac{\eta_{1,k}}{2} G_k^T G_k \right) \Delta \boldsymbol{x}_k \\
&\leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \frac{\eta_{2,k} \xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 + 3\eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \Delta \boldsymbol{x}_k{}^T \left(B_k + \left(\frac{\eta_{1,k}}{2} - \frac{8}{\eta_{2,k} \xi_G} \right) G_k^T G_k \right) \Delta \boldsymbol{x}_k . \quad (24)
\end{aligned}$$

In order to bound the last two terms of the above inequality, we decompose $\Delta \boldsymbol{x}_k$ as $\Delta \boldsymbol{x}_k = \Delta \boldsymbol{v}_k + \Delta \boldsymbol{u}_k$ where $\Delta \boldsymbol{v}_k = G_k^T \Delta \bar{\boldsymbol{v}}_k$ for some $\Delta \bar{\boldsymbol{v}}_k$ and $\Delta \boldsymbol{u}_k$ satisfies $G_k \Delta \boldsymbol{u}_k = \boldsymbol{0}$. Then, we have $\|\Delta \boldsymbol{x}_k\|^2 = \|\Delta \boldsymbol{v}_k\|^2 + \|\Delta \boldsymbol{u}_k\|^2$ and $\|\Delta \boldsymbol{v}_k\|^2 = \|G_k^T \Delta \bar{\boldsymbol{v}}_k\|^2 \leq \Upsilon^2 \|\Delta \bar{\boldsymbol{v}}_k\|^2$. Thus, $\|G_k \Delta \boldsymbol{x}_k\|^2 = \|G_k \Delta \boldsymbol{v}_k\|^2 = \left\| G_k G_k^T \Delta \bar{\boldsymbol{v}}_k \right\|^2 \geq \xi_G^2 \|\Delta \bar{\boldsymbol{v}}_k\|^2 \geq (\xi_G^2 / \Upsilon^2) \|\Delta \boldsymbol{v}_k\|^2$. Assuming $\eta_{1,k} \geq 16/(\eta_{2,k} \xi_G)$ at the moment and using Assumption 3.2 and Cauchy-Schwarz inequality, we get

$$\begin{aligned}
&3\eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \Delta \boldsymbol{x}_k{}^T \left(B_k + \left(\frac{\eta_{1,k}}{2} - \frac{8}{\eta_{2,k} \xi_G} \right) G_k^T G_k \right) \Delta \boldsymbol{x}_k \\
&\quad = 3\eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \Delta \boldsymbol{u}_k^T B_k \Delta \boldsymbol{u}_k - 2 \Delta \boldsymbol{u}_k^T B_k \Delta \boldsymbol{v}_k - \Delta \boldsymbol{v}_k^T B_k \Delta \boldsymbol{v}_k - \left(\frac{\eta_{1,k}}{2} - \frac{8}{\eta_{2,k} \xi_G} \right) \|G_k \Delta \boldsymbol{x}_k\|^2 \\
&\quad \leq 3\eta_{2,k} \Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \xi_B \|\Delta \boldsymbol{u}_k\|^2 + 2\Upsilon \|\Delta \boldsymbol{u}_k\| \|\Delta \boldsymbol{v}_k\| + \Upsilon \|\Delta \boldsymbol{v}_k\|^2 - \left(\frac{\eta_{1,k}}{2} - \frac{8}{\eta_{2,k} \xi_G} \right) \frac{\xi_G^2}{\Upsilon^2} \|\Delta \boldsymbol{v}_k\|^2 \\
&\quad \leq \left(3\eta_{2,k} \Upsilon^2 - \xi_B \right) \|\Delta \boldsymbol{x}_k\|^2 + 2\Upsilon \|\Delta \boldsymbol{u}_k\| \|\Delta \boldsymbol{v}_k\| + (\xi_B + \Upsilon) \|\Delta \boldsymbol{v}_k\|^2 - \left(\frac{\eta_{1,k} \xi_G^2}{2\Upsilon^2} - \frac{8\xi_G}{\eta_{2,k} \Upsilon^2} \right) \|\Delta \boldsymbol{v}_k\|^2 .
\end{aligned}$$

Applying Young's inequality for the second term, we obtain

$$2\Upsilon \|\Delta \boldsymbol{u}_k\| \|\Delta \boldsymbol{v}_k\| \leq \frac{\xi_B}{2} \|\Delta \boldsymbol{u}_k\|^2 + \frac{2\Upsilon^2}{\xi_B} \|\Delta \boldsymbol{v}_k\|^2 \leq \frac{\xi_B}{2} \|\Delta \boldsymbol{x}_k\|^2 + \frac{2\Upsilon^2}{\xi_B} \|\Delta \boldsymbol{v}_k\|^2 .$$

Combining the above two displays, we get

$$
\begin{aligned}
3\eta_{2,k}\Upsilon^2 \|\Delta \boldsymbol{x}_k\|^2 - \Delta \boldsymbol{x}_k{}^T \left(B_k + \left(\frac{\eta_{1,k}}{2} - \frac{8}{\eta_{2,k}\xi_G}\right) G_k^T G_k\right) \Delta \boldsymbol{x}_k \\
&\leq \left(3\eta_{2,k}\Upsilon^2 - \frac{\xi_B}{2}\right) \|\Delta \boldsymbol{x}_k\|^2 + \frac{2\Upsilon^2}{\xi_B} \|\Delta \boldsymbol{v}_k\|^2 + (\xi_B + \Upsilon) \|\Delta \boldsymbol{v}_k\|^2 - \left(\frac{\eta_{1,k}\xi_G^2}{2\Upsilon^2} - \frac{8\xi_G}{\eta_{2,k}\Upsilon^2}\right) \|\Delta \boldsymbol{v}_k\|^2 \\
&= \left(3\eta_{2,k}\Upsilon^2 - \frac{\xi_B}{2}\right) \|\Delta \boldsymbol{x}_k\|^2 + \left(\frac{2\Upsilon^2}{\xi_B} + \xi_B + \Upsilon + \frac{8\xi_G}{\eta_{2,k}\Upsilon^2} - \frac{\eta_{1,k}\xi_G^2}{2\Upsilon^2}\right) \|\Delta \boldsymbol{v}_k\|^2 .
\end{aligned}
$$

Combining the above inequality with (24), we get

$$
\begin{aligned}
\begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 + \left(3\eta_{2,k}\Upsilon^2 - \frac{\xi_B}{2}\right) \|\Delta \boldsymbol{x}_k\|^2 \\
+ \left(\frac{2\Upsilon^2}{\xi_B} + \xi_B + \Upsilon + \frac{8\xi_G}{\eta_{2,k}\Upsilon^2} - \frac{\eta_{1,k}\xi_G^2}{2\Upsilon^2}\right) \|\Delta \boldsymbol{v}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 .
\end{aligned}
$$

In order to make the second term on the right-hand side negative, we let

$$
\eta_{2,k} \leq \frac{\xi_B}{12\Upsilon^2}. \tag{25}
$$

Without loss of generality, we assume $\Upsilon/2 \geq 1 \geq (\xi_B \vee \xi_G)$. Otherwise, we replace Υ by $\Upsilon \vee 2$, ξ_B by $\xi_B \wedge 1$, and ξ_G by $\xi_G \wedge 1$. Then, we obtain

$$
\frac{2\Upsilon^2}{\xi_B} + \xi_B + \Upsilon + \frac{8\xi_G}{\eta_{2,k}\Upsilon^2} \leq \frac{2\Upsilon^2}{\xi_B} + \frac{3\Upsilon}{2} + \frac{8\xi_G}{\eta_{2,k}\Upsilon^2} \leq \frac{3\Upsilon^2}{\xi_B} + \frac{8\xi_G}{\eta_{2,k}\Upsilon^2} \overset{(25)}{\leq} \frac{1}{4\eta_{2,k}} + \frac{8\xi_G}{\eta_{2,k}\Upsilon^2} \leq \frac{1}{4\eta_{2,k}} + \frac{2}{\eta_{2,k}} \leq \frac{2.5}{\eta_{2,k}}.
$$

Using the above inequality and (25), we have

$$
\begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \frac{\xi_B}{4} \|\Delta \boldsymbol{x}_k\|^2 + \left(\frac{2.5}{\eta_{2,k}} - \frac{\eta_{1,k}\xi_G^2}{2\Upsilon^2}\right) \|\Delta \boldsymbol{v}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 . \tag{26}
$$

In order to make the right-hand side negative, we let

$$
\eta_{1,k} \geq \frac{5\Upsilon^2}{\eta_{2,k}\xi_G^2}. \tag{27}
$$

Notice that (25) and (27) imply $\eta_{1,k} \geq \eta_{2,k}$ and $\eta_{1,k} \geq 16/(\eta_{2,k}\xi_G)$, hence, justify our presumptions. From (26), we get

$$
\begin{aligned}
\begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} &\overset{(27)}{\leq} -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 - \frac{\xi_B}{4} \|\Delta \boldsymbol{x}_k\|^2 \\
&\overset{(25)}{\leq} -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8} \|\Delta \boldsymbol{\lambda}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8} \|\Delta \boldsymbol{x}_k\|^2 \\
&\leq -\frac{\eta_{2,k}}{2} \|\nabla \mathcal{L}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8} \|\Delta \boldsymbol{z}_k\|^2.
\end{aligned} \tag{28}
$$

Now we develop the second term in (23). By Cauchy-Schwarz inequality, we get

$$
\begin{aligned}
\begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \tilde{\Delta} \boldsymbol{x}_{k,j} - \Delta \boldsymbol{x}_k \\ \tilde{\Delta} \boldsymbol{\lambda}_{k,j} - \Delta \boldsymbol{\lambda}_k \end{pmatrix} &\overset{(2)}{=} \begin{pmatrix} \tilde{\Delta} \boldsymbol{x}_{k,j} - \Delta \boldsymbol{x}_k \\ \tilde{\Delta} \boldsymbol{\lambda}_{k,j} - \Delta \boldsymbol{\lambda}_k \end{pmatrix}^T \begin{pmatrix} I + \eta_{2,k} H_k & \eta_{1,k} G_k^T \\ \eta_{2,k} G_k & I \end{pmatrix} \begin{pmatrix} \nabla_{\boldsymbol{x}} \mathcal{L}_k \\ c_k \end{pmatrix} \\
&\overset{(4)}{=} -\begin{pmatrix} \tilde{\Delta} \boldsymbol{x}_{k,j} - \Delta \boldsymbol{x}_k \\ \tilde{\Delta} \boldsymbol{\lambda}_{k,j} - \Delta \boldsymbol{\lambda}_k \end{pmatrix}^T \begin{pmatrix} (I + \eta_{2,k} H_k) B_k + \eta_{1,k} G_k^T G_k & (I + \eta_{2,k} H_k) G_k^T \\ G_k (I + \eta_{2,k} B_k) & \eta_{2,k} G_k G_k^T \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \\
&\leq \left\| \begin{pmatrix} \tilde{\Delta} \boldsymbol{x}_{k,j} - \Delta \boldsymbol{x}_k \\ \tilde{\Delta} \boldsymbol{\lambda}_{k,j} - \Delta \boldsymbol{\lambda}_k \end{pmatrix} \right\| \left\| \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \right\| \left((1 + \eta_{2,k}\Upsilon)\Upsilon + (\eta_{1,k} + \eta_{2,k})\Upsilon^2 + 2(1 + \eta_{2,k}\Upsilon)\Upsilon\right) \\
&= \left\| \begin{pmatrix} \tilde{\Delta} \boldsymbol{x}_{k,j} - \Delta \boldsymbol{x}_k \\ \tilde{\Delta} \boldsymbol{\lambda}_{k,j} - \Delta \boldsymbol{\lambda}_k \end{pmatrix} \right\| \left\| \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \right\| \left(3\Upsilon + 4\eta_{2,k}\Upsilon^2 + \eta_{1,k}\Upsilon^2\right) \\
&\overset{(25)}{\leq} \left\| \begin{pmatrix} \tilde{\Delta} \boldsymbol{x}_{k,j} - \Delta \boldsymbol{x}_k \\ \tilde{\Delta} \boldsymbol{\lambda}_{k,j} - \Delta \boldsymbol{\lambda}_k \end{pmatrix} \right\| \left\| \begin{pmatrix} \Delta \boldsymbol{x}_k \\ \Delta \boldsymbol{\lambda}_k \end{pmatrix} \right\| \left(3\Upsilon + \xi_B/3 + \eta_{1,k}\Upsilon^2\right) .
\end{aligned}
$$

Using $\Upsilon/2 \geq 1 \geq (\xi_B \vee \xi_G)$ and (25), we get $\eta_{1,k} \overset{(27)}{\geq} (5\Upsilon^2)/(\eta_{2,k}\xi_G^2) \geq 20/\eta_{2,k} \geq 40 \cdot 24\Upsilon$. Then, we have $19/6 \leq \eta_{1,k}\Upsilon$ and further obtain

$$3\Upsilon + \frac{\xi_B}{3} + \eta_{1,k}\Upsilon^2 \leq \frac{19\Upsilon}{6} + \eta_{1,k}\Upsilon^2 \leq 2\eta_{1,k}\Upsilon^2.$$

Using the above inequality, we get

$$\begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} - \Delta\boldsymbol{x}_k \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} - \Delta\boldsymbol{\lambda}_k \end{pmatrix} \leq 2\eta_{1,k}\Upsilon^2 \left\| \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} - \Delta\boldsymbol{x}_k \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} - \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\| \left\| \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\|. \tag{29}$$

Finally, we obtain

$$\begin{aligned}
\begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} \end{pmatrix} &\overset{(23)}{=} \begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} + \begin{pmatrix} \nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k \end{pmatrix}^T \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} - \Delta\boldsymbol{x}_k \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} - \Delta\boldsymbol{\lambda}_k \end{pmatrix} \\
&\overset{(28)}{\underset{(29)}{\leq}} -\frac{\eta_{2,k}}{2}\|\nabla\mathcal{L}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8}\left\| \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\|^2 + 2\eta_{1,k}\Upsilon^2 \left\| \begin{pmatrix} \tilde{\Delta}\boldsymbol{x}_{k,j} - \Delta\boldsymbol{x}_k \\ \tilde{\Delta}\boldsymbol{\lambda}_{k,j} - \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\| \left\| \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\| \\
&\overset{(22)}{\leq} -\frac{\eta_{2,k}}{2}\|\nabla\mathcal{L}_k\|^2 - \frac{\eta_{2,k}\xi_G}{8}\left\| \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\|^2 + 2\delta_k\eta_{1,k}\Upsilon^2 \left\| \begin{pmatrix} \Delta\boldsymbol{x}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} \right\|^2 \\
&\leq -\frac{\eta_{2,k}}{2}\|\nabla\mathcal{L}_k\|^2 - \left(\frac{\eta_{2,k}\xi_G}{8} - 2\delta_k\eta_{1,k}\Upsilon^2\right)\|\Delta\boldsymbol{z}_k\|^2.
\end{aligned}$$

In order to make the right-hand side negative, we let

$$\delta_k \leq \frac{\eta_{2,k}\xi_G}{16\eta_{1,k}\Upsilon^2}.$$

Thus, the descent direction condition is satisfied as long as $\eta_{2,k} \leq \xi_B/12\Upsilon^2$, $\eta_{1,k}\eta_{2,k} \geq 5\Upsilon^2/\xi_G^2$, and $\delta_k\eta_{1,k}/\eta_{2,k} \leq \xi_G/(16\Upsilon^2)$. Finally, we let $\Upsilon \leftarrow 12\Upsilon^2/\xi_B \vee 5\Upsilon^2/\xi_G^2 \vee 16\Upsilon^2/\xi_G$, and complete the proof.

A.5. Proof of Lemma 3.8

We denote the event that the algorithm reaches $\boldsymbol{z}_k$ as $\mathcal{Z}_k$. Then the event $\cap_{k=0}^{\infty}\mathcal{Z}_k$ implies that the algorithm generates the iterates infinitely. From Lemmas 3.6 and 3.7, and noting that the Armijo condition (13) can be satisfied for small enough α_k as long as $\tilde{\Delta}\boldsymbol{z}_k$ is a descent direction (as implied by Lemma 3.7), we have $P(\mathcal{Z}_{k+1}|\mathcal{Z}_k) = 1, \forall k \geq 0$. Thus,

$$P\left(\cap_{k=0}^{\infty}\mathcal{Z}_k\right) \geq 1 - \sum_{k=0}^{\infty} P(\mathcal{Z}_k^c) = 1 - \sum_{k=0}^{\infty}\int P(\mathcal{Z}_{k+1}^c|\mathcal{Z}_k)P(\mathcal{Z}_k)d\boldsymbol{z}_k = 1. \tag{30}$$

We now start from finding the lower bound of $\delta_k^{\text{trial}}\eta_{1,k}/\eta_{2,k}$. Since the updating rule of the parameters (12) increases $\eta_{1,k}$ by a factor of ν^2 and decreases $\eta_{2,k}$ by a factor of ν, we know that $\eta_{1,0} \leq \eta_{1,k}$ and $\eta_{2,0} \geq \eta_{2,k}$ for all $k \geq 0$. By Assumptions 3.1 and 3.2, we know there exist constants $\Upsilon_\upsilon, \Psi > 0$ such that $\Upsilon_k \leq \Upsilon_\upsilon$ and $\Psi_k \leq \Psi$, $\forall k \geq 0$. Thus, for $k \geq 0$, we have

$$(1 + \eta_{1,k} + \eta_{2,k})\Upsilon_k^2\Psi_k^2 \leq \frac{\eta_{1,k}}{\eta_{1,0}}(\Upsilon_\upsilon^2\Psi^2 + \eta_{2,0}\Upsilon_\upsilon^2\Psi^2) + \eta_{1,k}\Upsilon_\upsilon^2\Psi^2 = \frac{\eta_{1,k}}{\eta_{1,0}}(\Upsilon_\upsilon^2\Psi^2 + \eta_{1,0}\Upsilon_\upsilon^2\Psi^2 + \eta_{2,0}\Upsilon_\upsilon^2\Psi^2).$$

Using the above inequality, we have

$$\frac{\delta_k^{\text{trial}}\eta_{1,k}}{\eta_{2,k}} \overset{(8)}{=} \frac{(0.5-\beta)\eta_{1,k}}{(1 + \eta_{1,k} + \eta_{2,k})\Upsilon_k^2\Psi_k^2} \geq \frac{(0.5-\beta)\eta_{1,0}}{\Upsilon_\upsilon^2\Psi^2 + \eta_{1,0}\Upsilon_\upsilon^2\Psi^2 + \eta_{2,0}\Upsilon_\upsilon^2\Psi^2}. \tag{31}$$

By Lemma 3.7 and (31), we have

$$\eta_{1,k}\eta_{2,k} \geq \Upsilon, \quad \eta_{2,k} \leq \frac{1}{\Upsilon}, \quad \frac{\delta_k\eta_{1,k}}{\eta_{2,k}} \leq \frac{1}{\Upsilon} \wedge \frac{(0.5-\beta)\eta_{1,0}}{\Upsilon_\upsilon^2\Psi^2 + \eta_{2,0}\Upsilon_\upsilon^2\Psi^2 + \eta_{1,0}\Upsilon_\upsilon^2\Psi^2}.$$

Notice that the lower bound of $\eta_{1,k}\eta_{2,k}$ and the upper bounds of $\eta_{2,k}$, $\delta_k\eta_{1,k}/\eta_{2,k}$ do not depend on k. Since the updating rule of the parameters (12) implies that $\eta_{1,k}\eta_{2,k}$ increases by a factor of ν, $\eta_{2,k}$ decreases by a factor of ν, and $\delta_k\eta_{1,k}/\eta_{2,k}$ decreases by at least a factor of ν, there exists an iteration threshold K such that $(\eta_{1,k}, \eta_{2,k}, \delta_k) = (\eta_{1,K}, \eta_{2,K}, \delta_K)$ for all k. Using (30), we complete the proof.

A.6. Proof of Lemma 3.9

By Assumption 3.1, we know that $\nabla\mathcal{L}_{\boldsymbol{\eta}}$ is Lipschitz continuous. Let us denote the Lipschitz constant of $\nabla\mathcal{L}_{\boldsymbol{\eta}_k}$ by Υ_{η_k}. We have for any $k \geq 0$,

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{z}_k + \alpha_k\tilde{\Delta}\boldsymbol{z}_k) &= \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k + \alpha_k\int_0^1\{\nabla\mathcal{L}^T_{\boldsymbol{\eta}_k}(\boldsymbol{z}_k + t\alpha_k\tilde{\Delta}\boldsymbol{z}_k) - \nabla\mathcal{L}^k_{\boldsymbol{\eta}_k}\}^T\tilde{\Delta}\boldsymbol{z}_k dt \\ &\leq \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k + \Upsilon_{\eta_k}\alpha_k^2\|\tilde{\Delta}\boldsymbol{z}_k\|^2\int_0^1 t dt \\ &= \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k + \Upsilon_{\eta_k}\alpha_k^2\|\tilde{\Delta}\boldsymbol{z}_k\|^2/2.\end{aligned}$$

By Assumption 3.2, we know there exists a constant $\Psi > 0$ such that $\Psi_k \leq \Psi$ for all $k \geq 0$. Using (22), we have

$$\|\tilde{\Delta}\boldsymbol{z}_k\| \leq \delta_k\|\Delta\boldsymbol{z}_k\| + \|\Delta\boldsymbol{z}_k\| = (\delta_k + 1)\|\Delta\boldsymbol{z}_k\| \leq 2\|\Delta\boldsymbol{z}_k\|. \tag{32}$$

By Lemma 3.8, we know that $\eta_{2,k} \geq \eta_{2,K}$ for all $k \geq 0$. We let $\Upsilon_\eta := \Upsilon_{\eta_0} \vee \cdots \vee \Upsilon_{\eta_K}$. Then, we have

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{z}_k + \alpha_k\tilde{\Delta}\boldsymbol{z}_k) &\leq \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k + \alpha_k^2\Upsilon_\eta\|\tilde{\Delta}\boldsymbol{z}_k\|^2/2 \\ &\overset{(32)}{\leq} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k + 2\alpha_k^2\Upsilon_\eta\|\Delta\boldsymbol{z}_k\|^2 \\ &\overset{(4)}{\underset{(20)}{\leq}} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k + 2\alpha_k^2\Upsilon_\eta\Psi^2\|\nabla\mathcal{L}_k\|^2 \\ &\overset{(11)}{\leq} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k - (4\alpha_k^2\Upsilon_\eta\Psi^2/\eta_{2,k})(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k \\ &\leq \mathcal{L}^k_{\boldsymbol{\eta}_k} + \alpha_k(1 - 4\alpha_k\Upsilon_\eta\Psi^2/\eta_{2,K})(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k.\end{aligned}$$

Now, we let

$$1 - \frac{4\alpha_k\Upsilon_\eta\Psi^2}{\eta_{2,K}} \geq \beta \Leftrightarrow \alpha_k \leq \frac{(1-\beta)\eta_{2,K}}{4\Upsilon_\eta\Psi^2}.$$

Since the upper bound of α_k does not depend on k, there exists $\alpha_{\min} > 0$ such that for any k, $\alpha_{\min} \leq \alpha_k$, when we do, for example, backtracking. Using (30), we complete the proof.

A.7. Proof of Theorem 3.10

By Lemmas 3.7, 3.8, and 3.9, we have for any $k \geq K$,

$$\mathcal{L}^{k+1}_{\boldsymbol{\eta}_K} - \mathcal{L}^k_{\boldsymbol{\eta}_K} \leq \alpha_k\beta(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_K})^T\tilde{\Delta}\boldsymbol{z}_k \overset{(11)}{\leq} -\eta_{2,k}\alpha_k\beta\|\nabla\mathcal{L}_k\|^2/2 \leq -\eta_{2,K}\alpha_{\min}\beta\|\nabla\mathcal{L}_k\|^2/2.$$

Summing over $k \geq K$, we have

$$\sum_{k=K}^{\infty}\|\nabla\mathcal{L}_k\|^2 \leq \frac{2}{\eta_{2,K}\alpha_{\min}\beta}\left(\mathcal{L}^K_{\boldsymbol{\eta}_K} - \min_{\mathcal{X}\times\Lambda}\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda})\right) < \infty.$$

Therefore, $\|\nabla\mathcal{L}_k\| \to 0$ as $k \to \infty$. Using (30), we have $P(\|\nabla\mathcal{L}_k\| \to 0 \text{ as } k \to \infty) = 1$. This completes the proof.

B. Proofs of Section 4

B.1. Proof of Theorem 4.3

We first show that for all sufficiently large k, a unit stepsize is admissible. It suffices to show that for all sufficiently large k,

$$\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{z}_k + \tilde{\Delta}\boldsymbol{z}_k) \leq \mathcal{L}^k_{\boldsymbol{\eta}_k} + \beta(\nabla\mathcal{L}^k_{\boldsymbol{\eta}_k})^T\tilde{\Delta}\boldsymbol{z}_k.$$

Let $\boldsymbol{u} \in \mathbb{R}^l$, $M : \mathbb{R}^l \to \mathbb{R}^{p\times q}$ and $w : \mathbb{R}^l \to \mathbb{R}^q$. Let us denote the columns of $M(\boldsymbol{u})$ as $m_i(\boldsymbol{u}) \in \mathbb{R}^p$ for $i = 1, \ldots, q$ and the entries of $w(\boldsymbol{u})$ as $w_i(\boldsymbol{u})$ for $i = 1, \ldots, q$. We can write $\nabla_{\boldsymbol{u}}(M(\boldsymbol{u})w(\boldsymbol{u})) \in \mathbb{R}^{l\times p}$ as

$$\nabla_{\boldsymbol{u}}(M(\boldsymbol{u})w(\boldsymbol{u})) = \sum_{i=1}^q w_i(\boldsymbol{u})\nabla_{\boldsymbol{u}}m_i(\boldsymbol{u}) + \nabla_{\boldsymbol{u}}w(\boldsymbol{u})M(\boldsymbol{u})^T =: \langle w(\boldsymbol{u}), \nabla_{\boldsymbol{u}}M(\boldsymbol{u})\rangle + \nabla_{\boldsymbol{u}}w(\boldsymbol{u})M(\boldsymbol{u})^T.$$

With the above definition and using (2), we can compute

$$\begin{aligned}
\nabla^2_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k} &= \nabla_{\boldsymbol{x}}\left(\nabla_{\boldsymbol{x}}\mathcal{L}_k + \eta_{2,k}H_k\nabla_{\boldsymbol{x}}\mathcal{L}_k + \eta_{1,k}G_k^T c_k\right) = H_k + \eta_{2,k}\left(\langle\nabla_{\boldsymbol{x}}\mathcal{L}_k, \nabla_{\boldsymbol{x}}H_k\rangle + H_k^2\right) + \eta_{1,k}\left(\langle c_k, \nabla G_k\rangle + G_k^T G_k\right),\\
\nabla^2_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k} &= \nabla_{\boldsymbol{\lambda}}\left(c_k + \eta_{2,k}G_k\nabla_{\boldsymbol{x}}\mathcal{L}_k\right) = \eta_{2,k}G_kG_k^T,\\
\nabla^2_{\boldsymbol{\lambda}\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k} &= \nabla_{\boldsymbol{\lambda}}\left(\nabla_{\boldsymbol{x}}\mathcal{L}_k + \eta_{2,k}H_k\nabla_{\boldsymbol{x}}\mathcal{L}_k + \eta_{1,k}G_k^T c^k\right) = G_k + \eta_{2,k}\left(\langle\nabla_{\boldsymbol{x}}\mathcal{L}_k, \nabla_{\boldsymbol{\lambda}}H_k\rangle + G_kH_k\right).
\end{aligned}$$

By Assumption 4.1, we know $\nabla^2\mathcal{L}_{\boldsymbol{\eta}}$ is continuous over $\mathcal{X}$. We define

$$\mathcal{H}_k = \begin{pmatrix} H_k + \eta_{2,k}H_k^2 + \eta_{1,k}G_k^TG_k & G_k^T + \eta_{2,k}H_kG_k^T \\ G_k + \eta_{2,k}G_kH_k & \eta_{2,k}G_kG_k^T \end{pmatrix}.$$

Using $\|\nabla\mathcal{L}_k\| = \|(\nabla_{\boldsymbol{x}}\mathcal{L}_k, c_k)\| = o(1)$, we have $\nabla^2\mathcal{L}^k_{\boldsymbol{\eta}_k} = \mathcal{H}_k + o(1)$. We now apply Taylor's theorem and obtain

$$\begin{aligned}
&\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{x}_k + \tilde{\Delta}\boldsymbol{x}_k, \boldsymbol{\lambda}_k + \tilde{\Delta}\boldsymbol{\lambda}_k)\\
&\quad\le \mathcal{L}^k_{\boldsymbol{\eta}_k} + \begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\nabla^2\mathcal{L}^k_{\boldsymbol{\eta}_k}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad= \mathcal{L}^k_{\boldsymbol{\eta}_k} + \begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\mathcal{H}_k\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad\overset{(2)}{\underset{(4)}{=}} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\mathcal{H}_k\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\\
&\qquad- \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\begin{pmatrix}(I+\eta_{2,k}H_k)B_k + \eta_{1,k}G_k^TG_k & (I+\eta_{2,k}H_k)G_k^T\\ G_k(I+\eta_{2,k}B_k) & \eta_{2,k}G_kG_k^T\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix} + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad= \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\mathcal{H}_k\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k - \Delta\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k - \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&\qquad+ \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\left(\mathcal{H}_k - \begin{pmatrix}(I+\eta_{2,k}H_k)B_k + \eta_{1,k}G_k^TG_k & (I+\eta_{2,k}H_k)G_k^T\\ G_k(I+\eta_{2,k}B_k) & \eta_{2,k}G^kG_k^T\end{pmatrix}\right)\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix} + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad= \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\mathcal{H}_k\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k - \Delta\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k - \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\
&\qquad+ \frac{1}{2}\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}^T\begin{pmatrix}(I+\eta_{2,k}H_k)(H_k - B_k) & \mathbf{0}\\ \eta_{2,k}G_k(H_k - B_k) & \mathbf{0}\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix} + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right).
\end{aligned}$$

By Assumption 4.2, we have $\|(H_k - B_k)\Delta\boldsymbol{x}_k\| \le \|(H_k - B_k)\|\,\|\Delta\boldsymbol{x}_k\| = o\left(\|\Delta\boldsymbol{x}_k\|\right)$. Thus, we have for any $k \ge 0$,

$$\begin{aligned}
&\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{x}_k + \tilde{\Delta}\boldsymbol{x}_k, \boldsymbol{\lambda}_k + \tilde{\Delta}\boldsymbol{\lambda}_k)\\
&\quad\le \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|\|\mathcal{H}_k\|\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k - \Delta\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k - \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|\\
&\qquad+ \frac{1}{2}\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|\left(\|I+\eta_{2,k}H_k\|\,\|(H_k - B_k)\Delta\boldsymbol{x}_k\| + \|\eta_{2,k}G_k\|\,\|(H_k - B_k)\Delta\boldsymbol{x}_k\|\right) + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad\overset{(22)}{\underset{(32)}{\le}} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \frac{1}{2}\|\mathcal{H}_k\|\cdot 2\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|\cdot\delta_k\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\| + o\left(\left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad\overset{(9)}{\le} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + \delta_k(3\Upsilon_k + \eta_{1,k}\Upsilon_k^2 + 4\eta_{2,k}\Upsilon_k^2)\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2 + o\left(\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right)\\
&\quad\overset{(9)}{\le} \mathcal{L}^k_{\boldsymbol{\eta}_k} + \frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}^k_{\boldsymbol{\eta}_k}\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + 4\delta_k\Upsilon_k^2(1 + \eta_{1,k} + \eta_{2,k})\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2 + o\left(\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right).
\end{aligned}\tag{33}$$

Using the fact that $\delta_k \le \delta_k^{\text{trial}} = \dfrac{(0.5-\beta)\eta_{2,k}}{(1+\eta_{1,k}+\eta_{2,k})\Upsilon_k^2\Psi_k^2}$, we have for any $k \ge 0$,

$$4.05\delta_k\Upsilon_k^2(1+\eta_{1,k}+\eta_{2,k})\|\Delta \boldsymbol{z}_k\|^2 \le (0.5-\beta)\,\frac{4.05\eta_{2,k}}{\Psi_k^2}\|\Delta \boldsymbol{z}_k\|^2 \overset{(4)}{\le} (0.5-\beta)\,\frac{\eta_{2,k}}{2}\frac{8.1}{\Psi_k^2}\|\Gamma_k^{-1}\|^2\,\|\nabla\mathcal{L}_k\|^2$$
$$\overset{(20)}{\le} (0.5-\beta)\,\eta_{2,k}\,\|\nabla\mathcal{L}_k\|^2/2 \overset{(11)}{\le} -(0.5-\beta)\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}.$$

Thus, for any $k \ge 0$, we have

$$\frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + 4.05\delta_k\Upsilon_k^2(1+\eta_{1,k}+\eta_{2,k})\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2 \le \beta\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}.$$

Thus, there exists an iteration threshold K_1 such that for any $k \ge K_1$, we have

$$\frac{1}{2}\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} + 4\delta_k\Upsilon_k^2(1+\eta_{1,k}+\eta_{2,k})\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2 + o\left(\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|^2\right) \le \beta\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix}.$$

Plugging the above inequality back into (33), we know for any $k \ge K_1$,

$$\mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{x}_k+\tilde{\Delta}\boldsymbol{x}_k, \boldsymbol{\lambda}_k+\tilde{\Delta}\boldsymbol{\lambda}_k) \le \mathcal{L}_{\boldsymbol{\eta}_k}^k + \beta\begin{pmatrix}\nabla_{\boldsymbol{x}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}_{\boldsymbol{\eta}_k}^k\end{pmatrix}^T\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k\end{pmatrix} \Leftrightarrow \mathcal{L}_{\boldsymbol{\eta}_k}(\boldsymbol{z}_k+\tilde{\Delta}\boldsymbol{z}_k) \le \mathcal{L}_{\boldsymbol{\eta}_k}^k + \beta(\nabla\mathcal{L}_{\boldsymbol{\eta}_k}^k)^T\tilde{\Delta}\boldsymbol{z}_k.$$

This completes the first part of the proof. Next, we show for all sufficiently large k,

$$\|\boldsymbol{z}_k+\tilde{\Delta}\boldsymbol{z}_k-\boldsymbol{z}^\star\| \le (1+\varphi)\theta\delta_K\|\boldsymbol{z}_k-\boldsymbol{z}^\star\|, \quad \text{for any } \varphi > 0.$$

We start from dividing $\boldsymbol{z}_k+\tilde{\Delta}\boldsymbol{z}_k-\boldsymbol{z}^\star$ into two terms as follows:

$$\boldsymbol{z}_k+\tilde{\Delta}\boldsymbol{z}_k-\boldsymbol{z}^\star = \begin{pmatrix}\boldsymbol{x}_k+\tilde{\Delta}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\tilde{\Delta}\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} = \begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} + \begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k-\Delta\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k-\Delta\boldsymbol{\lambda}_k\end{pmatrix}. \tag{34}$$

For the first term, we apply Assumption 3.2 (Γ_k is invertible) and have for any $k \ge 0$,

$$\begin{aligned}\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} &= \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} + \begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\\ &\overset{(4)}{=} \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} - \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\nabla\mathcal{L}_k\\ &= \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\left(\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} - \nabla\mathcal{L}_k\right)\\ &\overset{(3)}{=} \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\left(\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} - (\nabla\mathcal{L}_k-\nabla\mathcal{L}_\star)\right).\end{aligned} \tag{35}$$

By Assumption 3.1, we know $\nabla^2\mathcal{L}$ is continuous over $\mathcal{X}$. We apply Taylor's theorem and obtain

$$\begin{aligned}\nabla\mathcal{L}_k-\nabla\mathcal{L}_\star &= \int_0^1 \nabla^2\mathcal{L}\left(\boldsymbol{x}_k+t(\boldsymbol{x}^\star-\boldsymbol{x}_k), \boldsymbol{\lambda}_k+t(\boldsymbol{\lambda}^\star-\boldsymbol{\lambda}_k)\right)\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}dt\\ &= \int_0^1 \begin{pmatrix}H\left(\boldsymbol{x}_k+t(\boldsymbol{x}^\star-\boldsymbol{x}_k), \boldsymbol{\lambda}_k+t(\boldsymbol{\lambda}^\star-\boldsymbol{\lambda}_k)\right) & G^T(\boldsymbol{x}_k+t(\boldsymbol{x}^\star-\boldsymbol{x}_k))\\ G(\boldsymbol{x}_k+t(\boldsymbol{x}^\star-\boldsymbol{x}_k)) & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}dt\\ &:= \int_0^1 \begin{pmatrix}H_k(t) & G_k^T(t)\\ G_k(t) & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}dt.\end{aligned}$$

Plugging the above equation back into (35), we have

$$\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix} = \begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\left(\int_0^1 \begin{pmatrix}B_k-H_k(t) & G_k^T-G_k(t)^T\\ G_k-G_k(t) & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}dt\right).$$

By Assumption 3.1, we know H and G are Lipschitz continuous over $\mathcal{X}$. Thus, we apply Assumption 4.2 and have

$$\begin{aligned}\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\| &\le \left\|\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\right\|\left\|\int_0^1\begin{pmatrix}B_k-H_k(t) & G_k^T-G_k(t)^T\\ G_k-G_k(t) & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}dt\right\|\\ &\le \left\|\begin{pmatrix}B_k & G_k^T\\ G_k & \mathbf{0}\end{pmatrix}^{-1}\right\|\int_0^1\left\|\begin{pmatrix}B_k-H_k(t) & G_k^T-G_k(t)^T\\ G_k-G_k(t) & \mathbf{0}\end{pmatrix}\right\|\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|dt\\ &\le O\left(\tau_k\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\right)+O\left(\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|^2\right).\end{aligned}\tag{36}$$

Taking ℓ_2 norm on both sides of (34), we have

$$\begin{aligned}\left\|\begin{pmatrix}\boldsymbol{x}_k+\tilde{\Delta}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\tilde{\Delta}\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\| &\le \left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k-\Delta\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k-\Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\overset{(21)}{\le}\theta\delta_k\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\\ &\le \theta\delta_k\left(\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\right)+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\\ &\overset{(36)}{\le}\theta\delta_k\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|+o\left(\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\right).\end{aligned}$$

This completes the proof.

B.2. Proof of Corollary 4.4

By the proof of Theorem 4.3, we know that $\alpha_k = 1$ for all $k \ge K_1$ regardless of the value of θ_k. Taking ℓ_2 norm on both sides of (34), we have for any $k \ge 0$,

$$\begin{aligned}\left\|\begin{pmatrix}\boldsymbol{x}_k+\tilde{\Delta}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\tilde{\Delta}\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\| &\le \left\|\begin{pmatrix}\tilde{\Delta}\boldsymbol{x}_k-\Delta\boldsymbol{x}_k\\ \tilde{\Delta}\boldsymbol{\lambda}_k-\Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\\ &\overset{(21)}{\le}\theta_k\delta_k\left\|\begin{pmatrix}\Delta\boldsymbol{x}_k\\ \Delta\boldsymbol{\lambda}_k\end{pmatrix}\right\|+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\\ &\le \theta_k\delta_k\left(\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\right)+\left\|\begin{pmatrix}\boldsymbol{x}_k+\Delta\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k+\Delta\boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\\ &\overset{(36)}{\le}\theta_k\delta_k\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|+O\left(\tau_k\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|\right)+O\left(\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|^2\right)\\ &= O(\theta_k\delta_k+\tau_k)\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|+O\left(\left\|\begin{pmatrix}\boldsymbol{x}_k-\boldsymbol{x}^\star\\ \boldsymbol{\lambda}_k-\boldsymbol{\lambda}^\star\end{pmatrix}\right\|^2\right).\end{aligned}$$

This completes the proof.

C. Additional Algorithms, Tables, and Figures

Algorithm 2 and Algorithm 3 use the ℓ_1 penalized merit function of the form $\phi_\pi(\boldsymbol{x}) = f(\boldsymbol{x}) + \pi\|c(\boldsymbol{x})\|_1$. *Termination Test 1*, *Termination Test 2*, *Model Reduction Condition*, and π_k^{trial} are referred to in Byrd et al. (2008). Furthermore, we set $\sigma = \tau(1-\epsilon)$, $\beta = \kappa_2 = \|\nabla\mathcal{L}_0\|_1/(\|c_0\|_1+1)\vee 1$ as in Byrd et al. (2008). Algorithm 4 uses the augmented Lagrangian function of the form $\mathcal{L}_\mu(\boldsymbol{x},\boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda}) + (\mu/2)\|c(\boldsymbol{x})\|^2$.

Figure 3, Table 2, and Figure 4 present additional results.

Algorithm 2 Byrd et al. (2008) with the ℓ_1 penalized merit function

Input: initial iterate $\boldsymbol{z}_0$; scalars $\xi_B, \pi_0, \beta, \kappa > 0$, $\kappa_1, \epsilon, \tau, \eta \in (0, 1)$;
for $k = 0, 1, 2, \ldots$ **do**
  Compute f_k, ∇f_k, c_k, G_k, H_k, and generate B_k;
  Set $\tilde{\Delta}\boldsymbol{z}_k \leftarrow \mathbf{0}$ and compute $\boldsymbol{r}_k$ by (7);
  while *Termination Test 1* AND *Termination Test 2* does not hold **do**
    Update $\tilde{\Delta}\boldsymbol{z}_k$ and $\boldsymbol{r}_k$ by GMRES;
  end while
  if *Termination Test 2* is satisfied and *Model Reduction Condition* does not hold **then**
    Set $\pi_k \leftarrow \pi_k^{\text{trial}} + 10^{-4}$;
  end if
  Select α_k to satisfy the Armijo condition and update the iterate by (14);
  Set $\pi_{k+1} \leftarrow \pi_k$;
end for

Algorithm 3 A modified Byrd et al. (2008) scheme with adaptive design

Input: initial iterate $\boldsymbol{z}_0$; scalars $\xi_B, \pi_0, \kappa_0 > 0$, $\nu > 1$, $\eta \in (0, 1)$;
for $k = 0, 1, 2, \ldots$ **do**
  Compute f_k, ∇f_k, c_k, G_k, H_k, and generate B_k;
  Set $\tilde{\Delta}\boldsymbol{z}_k \leftarrow \mathbf{0}$ and compute $\boldsymbol{r}_k$ by (7)
  while *Termination Test 1* does not hold **do**
    while $\|\boldsymbol{r}_k\|_1 > \kappa_k \|\nabla \mathcal{L}_k\|_1$ **do**
      Update $\tilde{\Delta}\boldsymbol{z}_k$ and $\boldsymbol{r}_k$ by GMRES;
    end while
    if *Model Reduction Condition* does not hold **then**
      Set $\pi_k \leftarrow \pi_k \nu$ and $\kappa_k \leftarrow \kappa_k / \nu^2$;
    end if
  end while
  Select α_k to satisfy the Armijo condition and update the iterate by (14);
  Set $\pi_{k+1} \leftarrow \pi_k$ and $\kappa_{k+1} \leftarrow \kappa_k$;
end for

Algorithm 4 Augmented Lagrangian method (Nocedal & Wright, 2006, Algorithm 17.3)

Input: initial iterate $\boldsymbol{z}_0 = (\boldsymbol{x}_0, \boldsymbol{\lambda}_0)$; scalars $\mu_0, \tau_0, \kappa > 0$, $\nu_\mu > 1$, $\nu_\tau \in (0, 1)$, $\eta \in (0, 1)$;
for $k = 0, 1, 2, \ldots$ **do**
  Set $\boldsymbol{x}_k^s \leftarrow \boldsymbol{x}_k$;
  while $\|\nabla_{\boldsymbol{x}} \mathcal{L}_{\mu_k}(\boldsymbol{x}_k^s, \boldsymbol{\lambda}_k)\| > \tau_k$ **do**
    Compute the gradient and modified Hessian of the augmented Lagrangian at $(\boldsymbol{x}_k^s, \boldsymbol{\lambda}_k)$;
    Find a search direction $\Delta \boldsymbol{x}_k^s$ via an inexact Newton subproblem solver with GMRES and a forcing term κ;
    Select α_k to satisfy the Armijo condition and set $\boldsymbol{x}_k^s \leftarrow \boldsymbol{x}_k^s + \alpha_k \Delta \boldsymbol{x}_k^s$;
    Update κ;
  end while
  Set $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k^s$ and $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k + \mu_k c_k$;
  Set $\mu_{k+1} \leftarrow \nu_\mu \mu_k$ and $\tau_{k+1} \leftarrow \nu_\tau \tau_k$;
end for

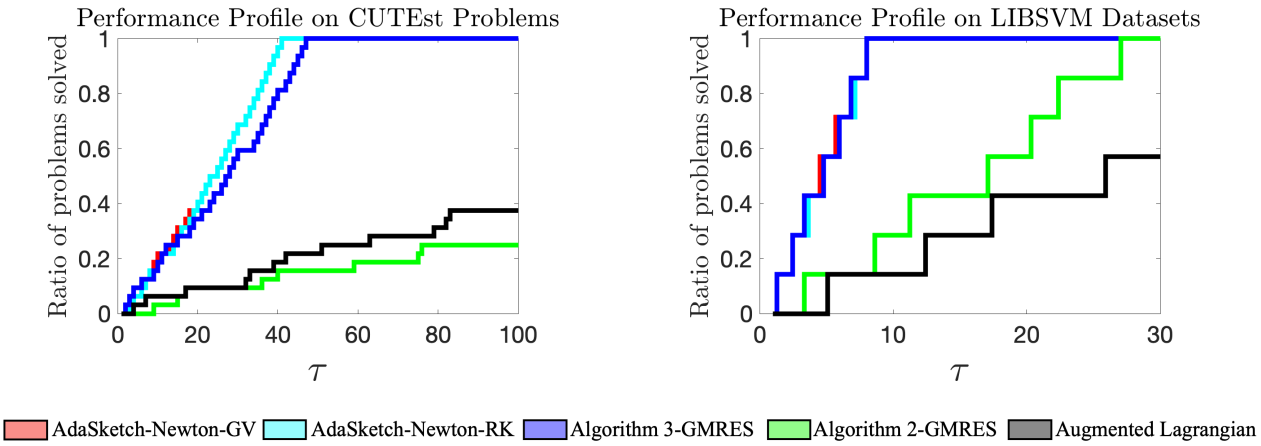

Figure 3: The performance profiles of the total number of flops for AdaSketch-Newton, Algorithms 2, Algorithm 3, and Augmented Lagrangian on CUTEst problems (Left) and on 7 LIBSVM datasets (right); the ratio of the problems solved on the y axis, while the proportion of the total number of flops (called the performance ratio and denoted as τ) on the x axis. See Dolan & Moré (2002) for more details.

Table 2: Numerical Results for PDE-constrained Problem

METHOD	KKT RESIDUAL	OBJ. AND CONS. EVAL	GRAD. AND JACOB. EVAL
ADASKETCH-NEWTON-GV	1.79E-5	18	10
ADASKETCH-NEWTON-RK	4.22E-6	14	8
ALGORITHM 3-GMRES	9.10E-5	42	12
ALGORITHM 2-GMRES	9.95E-5	130	34
AUGMENTED LAGRANGIAN	9.99E-5	278	73

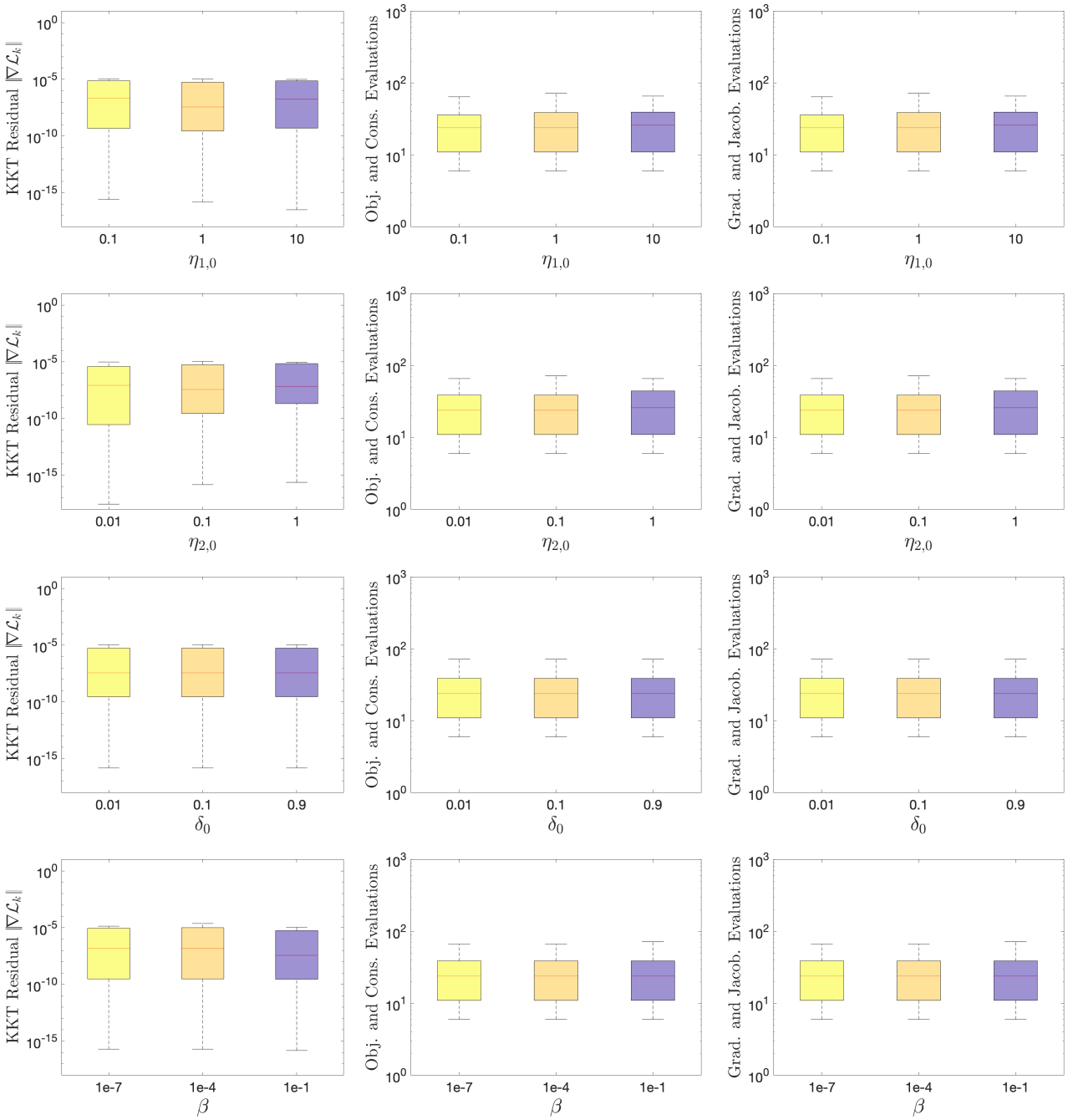

Figure 4: The boxplots of the KKT residual, the number of objective and constraints evaluations, and the number of gradient and Jacobian evaluations for AdaSketch-Newton-GV with different settings of the tuning parameters $(\eta_{1,0}, \eta_{2,0}, \delta_0, \beta)$ on CUTEst problems.