Online Learning Guided Curvature Approximation: A Quasi-Newton Method with Global Non-Asymptotic Superlinear Convergence

Ruichen Jiang RJIANG@UTEXAS.EDU
Qiujiang Jin QIUJIANG@AUSTIN.UTEXAS.EDU
Aryan Mokhtari MOKHTARI@AUSTIN.UTEXAS.EDU
The University of Texas at Austin

Editors: Gergely Neu and Lorenzo Rosasco

Abstract

Quasi-Newton algorithms are among the most popular iterative methods for solving unconstrained minimization problems, largely due to their favorable superlinear convergence property. However, existing results for these algorithms are limited as they provide either (i) a global convergence guarantee with an *asymptotic* superlinear convergence rate, or (ii) a *local* non-asymptotic superlinear rate for the case that the initial point and the initial Hessian approximation are chosen properly. In particular, no current analysis for quasi-Newton methods guarantees global convergence with an explicit superlinear convergence rate. In this paper, we close this gap and present the first *globally* convergent quasi-Newton method with an *explicit non-asymptotic* superlinear convergence rate. Unlike classical quasi-Newton methods, we build our algorithm upon the hybrid proximal extragradient method and propose a novel *online learning* framework for updating the Hessian approximation matrices. Specifically, guided by the convergence analysis, we formulate the Hessian approximation update as an online convex optimization problem in the space of matrices, and we relate the bounded regret of the online problem to the superlinear convergence of our method.

Keywords: Quasi-Newton methods, non-asymptotic superlinear convergence rate, online learning

1. Introduction

In this paper, we study quasi-Newton methods to solve unconstrained optimization problems. This class of algorithms can be viewed as a modification of Newton's method, where the objective function Hessian is approximated using the gradient information. Specifically, a general template of quasi-Newton methods to minimize a continuously differentiable function f is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \rho_k \mathbf{B}_k^{-1} \nabla f(\mathbf{x}_k), \qquad k \geq 0, \tag{1}$$

where ρ_k is the step size and $\mathbf{B}_k$ is a matrix that aims to approximate $\nabla^2 f(\mathbf{x}_k)$. Several rules for updating $\mathbf{B}_k$ have been proposed in the literature, and the most prominent include the Davidon-Fletcher-Powell (DFP) method (Davidon, 1959; Fletcher and Powell, 1963), the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970), and the symmetric rank-one (SR1) method (Conn et al., 1991; Khalfan et al., 1993).

The main advantage of quasi-Newton methods is their ability to achieve Q-superlinear convergence under suitable conditions on f, i.e., $\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1}-\mathbf{x}^*\|}{\|\mathbf{x}_k-\mathbf{x}^*\|} = 0$ where $\mathbf{x}^*$ is the optimal solution of f. Broyden et al. (1973); Dennis and Moré (1974) established that DFP and BFGS are locally and Q-superlinearly convergent with unit step size (i.e., $\rho_k = 1$ in (1)). To ensure global convergence, it is necessary to incorporate quasi-Newton updates with a line search or a trust-region method. Powell (1971); Dixon (1972) proved that DFP and BFGS converge globally and Q-superlinearly with an exact line search, which can be computationally prohibitive. Subsequently, Powell (1976) showed that BFGS with an inexact line search retains global and superlinear convergence, and Byrd et al. (1987) later extended the result to the restricted Broyden class except for DFP. Along another line of research, Conn et al. (1991); Khalfan et al. (1993); Byrd et al. (1996) studied the SR1 method in a trust region context and also proved its global and superlinear convergence. However, the above results are all of *asymptotic* nature and they fail to provide an explicit upper bound on the distance to the optimal solution after a finite number of iterations.

To address this shortcoming, several recent papers (Rodomanov and Nesterov, 2021b,a,c; Jin and Mokhtari, 2022; Jin et al., 2022; Lin et al., 2021; Ye et al., 2022) have studied *local nonasymptotic* superlinear convergence rates of classic quasi-Newton methods or their greedy variants. In particular, Rodomanov and Nesterov (2021c) proved that in a local neighborhood of the optimal solution, if the initial Hessian approximation is set as $L_1\mathbf{B}$ where $\mathbf{B}$ is a positive definite matrix, BFGS with unit step size converges at a superlinear rate of $[e^{\frac{d}{k}\log\frac{L_1}{\mu}}-1]^{k/2}$, where k is the number of iterations, d is the problem dimension, L_1 is the smoothness parameter and μ is the strong convexity parameter relative to the matrix $\mathbf{B}$ (see eq. (26) in (Rodomanov and Nesterov, 2021c)). Note that since L_1 and μ are defined with respect to $\mathbf{B}$, in general, this superlinear rate will depend on the condition number of $\mathbf{B}$. In a concurrent work, Jin and Mokhtari (2022) showed that if the initial Hessian approximation is close to the Hessian at the optimal solution or selected as the Hessian at the initial point, BFGS with unit step size can achieve a local superlinear convergence rate of $(1/k)^{k/2}$. However, all these results are crucially based on *local analysis*, and there is no clear way of extending these local non-asymptotic superlinear rates into global convergence guarantees for quasi-Newton methods.

Specifically, the existing local analyses in both (Rodomanov and Nesterov, 2021c) and (Jin and Mokhtari, 2022) require the initial point $\mathbf{x}_0$ to be close enough to the optimal solution $\mathbf{x}^*$, and in this local regime the step size in (1) has to be $\rho_k = 1$. Hence, to obtain a global convergence guarantee, it is necessary to use a globalization strategy, such as a line search scheme, and then switch to the local analysis when the iterates reach a local neighborhood of $\mathbf{x}^*$. However, this approach faces several challenges: (i) It is unclear how to obtain an explicit global convergence rate for quasi-Newton methods with line search. (ii) It is unclear how to bound the number of iterations before the line search scheme can accept the unit step size $\rho_k = 1$. (iii) Moreover, regarding the result in (Rodomanov and Nesterov, 2021c), it is unclear how to control the condition number of the Hessian approximation matrix when the iterates enter the local neighborhood, which would affect the region of local convergence and the starting moment of superlinear convergence. Similarly, to apply the local analysis in (Jin and Mokhtari, 2022), the Hessian approximation matrix need be close to the exact Hessian in a local neighborhood, which cannot be guaranteed in general. Hence, the following question remains open:

> *Can we design a globally convergent quasi-Newton method with an explicit superlinear convergence rate?*

In this paper, we answer the above question in affirmative. We propose a novel quasi-Newton proximal extragradient (QNPE) method that achieves an explicit non-asymptotic superlinear convergence rate. Unlike prior works that use a *local analysis* requiring specific conditions on the initial iterate and the initial Hessian approximation, our *global superlinear* convergence guarantee holds for an arbitrary initialization of the iterate and Hessian approximation. More precisely, for a μ-strongly convex function f with L_1-Lipschitz gradient and L_2-Lipschitz Hessian, the iterates $\{\mathbf{x}_k\}_{k\geq 0}$ generated by our QNPE method satisfy the following guarantees:

(i) Global convergence rates. We have $\frac{\|\mathbf{x}_k-\mathbf{x}^*\|^2}{\|\mathbf{x}_0-\mathbf{x}^*\|^2} \leq \min\{(1+\frac{\mu}{4L_1})^{-k}, (1+\frac{\mu}{4L_1}\sqrt{k/C})^{-k}\}$, where $C = \mathcal{O}\Big(\frac{\|\mathbf{B}_0-\nabla^2 f(\mathbf{x}^*)\|_F^2}{L_1^2} + \frac{L_2^2\|\mathbf{x}_0-\mathbf{x}^*\|^2}{\mu L_1}\Big) = \mathcal{O}\Big(d + \frac{L_2^2\|\mathbf{x}_0-\mathbf{x}^*\|^2}{\mu L_1}\Big)$. Note that the first bound corresponds to a linear convergence rate on par with the rate of gradient descent, while the second one corresponds to a superlinear rate. In particular, the superlinear rate outperforms the linear rate when $k \geq C$.

(ii) Iteration complexity. Let N_ϵ denote the number of iterations required to reach ϵ-accuracy. Then we have $N_\epsilon = \mathcal{O}\Big(\min\Big\{\frac{L_1}{\mu}\log\frac{1}{\epsilon}, \log\frac{1}{\epsilon}\Big/\log\Big(1+\frac{\mu}{L_1}\Big(\frac{L_1}{C\mu}\log\frac{1}{\epsilon}\Big)^{1/3}\Big)\Big\}\Big)$. In particular, in the regime where ϵ is sufficiently small, we obtain $N_\epsilon = \mathcal{O}\left(\log\frac{1}{\epsilon}/\log\log\frac{1}{\epsilon}\right)$.

(iii) Computational complexity. To achieve ϵ-accuracy, the total number of gradient evaluations and matrix-vector products is bounded by $3N_\epsilon - 1$ and $\mathcal{O}(N_\epsilon\sqrt{\frac{L_1}{\mu}}\log\frac{L_1 N_\epsilon^2 d}{\mu\epsilon})$, respectively.

We obtain these results by taking a quite different route from the existing quasi-Newton literature. Instead of considering an update of the form (1), we build our method based on the *hybrid proximal extragradient* (HPE) framework (Solodov and Svaiter, 1999), which can be interpreted as an inexact variant of the proximal point method (Martinet, 1970; Rockafellar, 1976). The general HPE method consists of two steps: an inexact proximal point update where $\hat{\mathbf{x}}_k \approx \mathbf{x}_k - \eta_k\nabla f(\hat{\mathbf{x}}_k)$, and an extragradient step where $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k\nabla f(\hat{\mathbf{x}}_k)$. In our QNPE method, we implement the first step by using the linear approximation $\nabla f(\mathbf{x}_k) + \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k)$ as a surrogate of $\nabla f(\hat{\mathbf{x}}_k)$. Considering this approximation and by exploiting strong convexity, the QNPE update is given by

$$\hat{\mathbf{x}}_k = \mathbf{x}_k - \eta_k(\mathbf{I}+\eta_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k), \quad \mathbf{x}_{k+1} = \frac{1}{1+2\eta_k\mu}(\mathbf{x}_k - \eta_k\nabla f(\hat{\mathbf{x}}_k)) + \frac{2\eta_k\mu}{1+2\eta_k\mu}\hat{\mathbf{x}}_k, \tag{2}$$

where η_k is the step size and μ is the strong convexity parameter.

Moreover, to ensure that QNPE preserves the fast convergence rate of HPE, we develop a novel scheme for the update of $\mathbf{B}_k$ to control the error caused by the linear approximation. As a result, unlike traditional quasi-Newton methods (such as BFGS and DFP) that update $\mathbf{B}_k$ by mimicking some property of the Hessian such as the secant condition, our update rule is directly motivated by the convergence analysis of the HPE framework. Specifically, according to our analysis, it is sufficient to ensure that $\sum_k 1/\eta_k^2 < +\infty$ in order to guarantee a superlinear convergence rate for the QNPE method. As we discuss later, this sum can be explicitly bounded above by the cumulative loss $\sum_k \ell_k(\mathbf{B}_k)$, where $\ell_k : \mathbb{S}_+^d \to \mathbb{R}_+$ is a loss function that in some sense measures the approximation error. As a result, the update of $\mathbf{B}_k$ boils down to running an online algorithm for solving an *online convex optimization problem* in the space of positive definite matrices with bounded eigenvalues.

Finally, we address the challenge of computational efficiency by presenting a projection-free online learning algorithm for the update of $\mathbf{B}_k$. Note that most online learning algorithms for constrained problems are based on a projection oracle, but in our specific setting, such projection requires expensive eigendecomposition to ensure that the eigenvalues of $\mathbf{B}_k$ are within a specific range. In contrast, our projection-free online learning algorithm is based on an approximate separation oracle (see Definition 7) that can be efficiently constructed using matrix-vector products.

2. Preliminaries

In this paper, we focus on the unconstrained minimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \quad f(\mathbf{x}), \tag{3}$$

where $f:\mathbb{R}^d \to \mathbb{R}$ is convex and twice differentiable and satisfies the following assumptions.

Assumption 1 *There exist positive constants μ and L_1 such that $\mu\mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq L_1\mathbf{I}$ for any $\mathbf{x}\in\mathbb{R}^d$, where $\mathbf{I}\in\mathbb{R}^{d\times d}$ is the identity matrix. That is, f is μ-strongly convex and L_1-smooth.*

Assumption 2 *There exists $L_2 > 0$ such that $\|\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{x}^*)\|_{\mathrm{op}} \leq L_2\|\mathbf{x}-\mathbf{x}^*\|_2$ for any $\mathbf{x}\in\mathbb{R}^d$, where $\mathbf{x}^*$ is the optimal solution and $\|\mathbf{A}\|_{\mathrm{op}} \triangleq \sup_{\mathbf{x}:\|\mathbf{x}\|_2=1}\|\mathbf{A}\mathbf{x}\|_2$.*

We note that both assumptions are standard; Assumption 1 is common in the study of first-order methods, while Assumption 2 is also used in various papers on the superlinear convergence of classical quasi-Newton methods (e.g., see Byrd et al. (1987); Jin and Mokhtari (2022)). For instance, both the regularized log-sum-exp function and the loss function of regularized logistic regression satisfy our assumptions (see Rodomanov and Nesterov (2021b)). Also, unless otherwise specified, throughout the paper we use $\|\cdot\|$ to denote the Euclidean norm.

Hybrid Proximal Extragradient Framework. To set the stage for our algorithm, we briefly recap the hybrid proximal extragradient (HPE) framework in (Solodov and Svaiter, 1999; Monteiro and Svaiter, 2010, 2012). When specialized to the minimization problem in (3), it can be described by the following two steps: First, we perform an inexact proximal point update $\hat{\mathbf{x}}_k \approx \mathbf{x}_k - \eta_k\nabla f(\hat{\mathbf{x}}_k)$ with step size η_k. More precisely, for a given parameter $\sigma\in[0,1)$, we find $\hat{\mathbf{x}}_k$ that satisfies

$$\|\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k\nabla f(\hat{\mathbf{x}}_k)\| \leq \sigma\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|. \tag{4}$$

Then, we perform an extragradient step and compute $\mathbf{x}_{k+1}$ by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k\nabla f(\hat{\mathbf{x}}_k). \tag{5}$$

While the proximal point update is only computed inexactly in (4), Monteiro and Svaiter (2010) proved that the HPE method can achieve a similar convergence guarantee as the proximal point method. Specifically, when f is convex, it holds that $f(\bar{\mathbf{x}}_{N-1}) - f(\mathbf{x}^*) \leq \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|^2(\sum_{k=0}^{N-1}\eta_k)^{-1}$, where $\bar{\mathbf{x}}_{N-1} \triangleq \sum_{k=0}^{N-1}\eta_k\hat{\mathbf{x}}_k / \sum_{k=0}^{N-1}\eta_k$ is the averaged iterate. It is worth noting that the HPE method is not directly implementable, but rather a useful conceptual tool, as we still need to specify how to find $\hat{\mathbf{x}}$ satisfying the condition in (4). One systematic approach is to approximate the gradient operator ∇f by a simpler local model $P(\mathbf{x};\mathbf{x}_k)$, and then compute $\hat{\mathbf{x}}_k$ by solving the equation

$$\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k P(\hat{\mathbf{x}}_k;\mathbf{x}_k) = 0. \tag{6}$$

Furthermore, we can see that the condition in (4) becomes

$$\eta_k\|\nabla f(\hat{\mathbf{x}}_k) - P(\hat{\mathbf{x}}_k;\mathbf{x}_k)\| \leq \sigma\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|, \tag{7}$$

which imposes an upper bound on the step size depending on the approximation error. For instance, if we take $P(\mathbf{x};\mathbf{x}_k) = \nabla f(\mathbf{x}_k)$, the update in (6) reads $\hat{\mathbf{x}}_k = \mathbf{x}_k - \eta_k\nabla f(\mathbf{x}_k)$, leading to the classic extragradient method by Korpelevich (1976). If we use $P(\mathbf{x};\mathbf{x}_k) = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k)$, we obtain the Newton proximal extragradient (NPE) method by Monteiro and Svaiter (2010, 2012),

Algorithm 1 Quasi-Newton Proximal Extragradient (QNPE) Method (informal)

```
Input: strong convexity parameter $\mu$, smoothness parameter $L_1$, line search parameters $\alpha_1 \geq 0$ and $\alpha_2 > 0$ such that $\alpha_1 + \alpha_2 < 1$, and initial trial step size $\sigma_0 > 0$
Initialization: initial point $\mathbf{x}_0 \in \mathbb{R}^d$ and initial Hessian approximation $\mathbf{B}_0$ such that $\mu\mathbf{I} \preceq \mathbf{B}_0 \preceq L_1\mathbf{I}$
for iteration $k = 0, \ldots, N-1$ do
    Let $\eta_k$ be the largest possible step size in $\{\sigma_k\beta^i : i \geq 0\}$ such that
        $\hat{\mathbf{x}}_k \approx_{\alpha_1} \mathbf{x}_k - \eta_k(\mathbf{I} + \eta_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k)$,    (see Eq. (8))
        $\eta_k\|\nabla f(\hat{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k)\| \leq \alpha_2\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|$.
    Set $\sigma_{k+1} \leftarrow \eta_k/\beta$
        } Line search subroutine; see Section 3.1
    Update $\mathbf{x}_{k+1} \leftarrow \frac{1}{1+2\eta_k\mu}(\mathbf{x}_k - \eta_k\nabla f(\hat{\mathbf{x}}_k)) + \frac{2\eta_k\mu}{1+2\eta_k\mu}\hat{\mathbf{x}}_k$
    if $\eta_k = \sigma_k$ then    # Line search accepted the initial trial step size
        Set $\mathbf{B}_{k+1} \leftarrow \mathbf{B}_k$
    else    # Line search bactracked
        Let $\tilde{\mathbf{x}}_k$ be the last rejected iterate in the line search
        Set $\mathbf{y}_k \leftarrow \nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k)$ and $\mathbf{s}_k \leftarrow \tilde{\mathbf{x}}_k - \mathbf{x}_k$
        Define the loss function $\ell_k(\mathbf{B}) = \frac{\|\mathbf{y}_k - \mathbf{B}\mathbf{s}_k\|^2}{2\|\mathbf{s}_k\|^2}$
        Feed $\ell_k(\mathbf{B})$ to an online learning algorithm and obtain $\mathbf{B}_{k+1}$
    end if
        } Hessian approximation update subroutine; see Section 3.2
end for
```

which has a faster convergence rate in the convex setting. However, the NPE method requires access to the objective function Hessian, which could be computationally costly. In this paper, we propose a quasi-Newton proximal extragradient method that only requires access to gradients. Surprisingly, our update rule for the Hessian approximation matrix does not follow traditional update rules such as the ones in BFGS or DFP, but is instead guided by an online learning approach, where we aim to minimize the regret corresponding to certain approximation error. More details are in Section 3.

3. Quasi-Newton Proximal Extragradient Method

In this section, we propose the quasi-Newton proximal extragradient (QNPE) method. An informal description is provided in Algorithm 1. On a high level, our method falls into the HPE framework described in Section 2. In particular, we choose the local model in (6) and (7) as $P(\mathbf{x}; \mathbf{x}_k) = \nabla f(\mathbf{x}_k) + \mathbf{B}_k(\mathbf{x} - \mathbf{x}_k)$, where $\mathbf{B}_k \in \mathbb{S}^d_+$ is the Hessian approximation matrix. Specifically, the update at the k-th iteration consists of three major stages, which we describe in the following paragraphs.

In the **first stage**, given the Hessian approximation matrix $\mathbf{B}_k$ and the current iterate $\mathbf{x}_k$, we select the step size η_k and the point $\hat{\mathbf{x}}_k$ such that

$$\|\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k(\nabla f(\mathbf{x}_k) + \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k))\| \leq \alpha_1\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|, \tag{8}$$

$$\eta_k\|\nabla f(\hat{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k)\| \leq \alpha_2\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|, \tag{9}$$

where $\alpha_1 \in [0, 1)$ and $\alpha_2 \in (0, 1)$ are user-specified parameters with $\alpha_1 + \alpha_2 < 1$. The first condition in (8) requires $\hat{\mathbf{x}}_k$ to be an inexact solution of the linear system of equations $(\mathbf{I} + \eta_k\mathbf{B}_k)(\mathbf{x} - \mathbf{x}_k) = -\eta_k\nabla f(\mathbf{x}_k)$, where α_1 controls the error of solving the linear system. In particular, when $\alpha_1 = 0$, it reduces to the update $\hat{\mathbf{x}}_k = \mathbf{x}_k - \eta_k(\mathbf{I} + \eta_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k)$ as in (2). The second condition in (9) ensures that the approximation error between the gradient $\nabla f(\hat{\mathbf{x}}_k)$ and its quasi-Newton approximation $\nabla f(\mathbf{x}_k) + \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k)$ is sufficiently small. To satisfy the conditions in (8) and (9) simultaneously, we need to determine the step size η_k and the point $\hat{\mathbf{x}}_k$ by a *line search subroutine* (cf. Lines 4-5 in Algorithm 1). Specifically, for a given parameter $\beta \in (0,1)$, we choose the largest admissible step size from the set $\{\sigma_k \beta^i : i \geq 0\}$, where $\sigma_k = \eta_{k-1}/\beta$ for $k \geq 1$. This can be implemented by a backtracking line search scheme and we present the details in Section 3.1.

In the **second stage**, we compute $\mathbf{x}_{k+1}$ using the gradient at $\hat{\mathbf{x}}_k$ (cf. Line 6 in Algorithm 1), but our update is slightly different from the one in (5) as we focus on the strongly-convex setting, while the update in (5) is designed for the convex setting. More precisely, we compute $\mathbf{x}_{k+1}$ according to

$$\mathbf{x}_{k+1} = \frac{1}{1+2\eta_k\mu}(\mathbf{x}_k - \eta_k \nabla f(\hat{\mathbf{x}}_k)) + \frac{2\eta_k\mu}{1+2\eta_k\mu}\hat{\mathbf{x}}_k, \tag{10}$$

where we choose the coefficients based on our analysis to obtain the best convergence rate. Note that the above update in (10) reduces to (5) when $\mu = 0$, and thus it can be viewed as an extension of HPE to the strongly-convex setting, which appears to be novel and of independent interest.

In the **third stage**, we update the Hessian approximation matrix $\mathbf{B}_k$. Here, we take a different approach from the classical quasi-Newton methods (such as BFGS and DFP) and let the convergence analysis guide our choice of $\mathbf{B}_{k+1}$. As will be evident later, the convergence rate of Algorithm 1 is closely related to the cumulative loss $\sum_{k\in\mathcal{B}} \ell_k(\mathbf{B}_k)$, where $\mathcal{B}$ denotes the set of iteration indices where the line search procedure backtracks. Here, the loss function is given by $\ell_k(\mathbf{B}_k) \triangleq \frac{\|\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\|^2}{2\|\mathbf{s}_k\|^2}$, where $\mathbf{y}_k = \nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k)$, $\mathbf{s}_k = \tilde{\mathbf{x}}_k - \mathbf{x}_k$, and $\tilde{\mathbf{x}}_k$ is an auxiliary iterate returned by our line search scheme. Thus, the update of the Hessian approximation matrix naturally fits into the framework of *online learning*. More precisely, if the line search accepts the initial trial step size (i.e., $k \notin \mathcal{B}$), we keep the Hessian approximation matrix unchanged (cf. Line 8 in Algorithm 1). Otherwise, we follow a tailored projection-free online learning algorithm in the space of matrices (cf. Line 13 in Algorithm 1). The details of the update of $\mathbf{B}_k$ are in Section 3.2.

Finally, we provide a convergence guarantee for QNPE in Proposition 1, which serves as a cornerstone for our convergence analysis. We note that the following result does not require additional conditions on $\mathbf{B}_k$, other than the ones in (8) and (9) . The proof is available in Appendix A.1.

Proposition 1 *Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the iterates generated by (8), (9), and (10) where $\alpha_1 + \alpha_2 < 1$. If f is μ-strongly convex, then $\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_k - \mathbf{x}^*\|^2(1+2\eta_k\mu)^{-1}$.*

Proposition 1 highlights the pivotal role of η_k in the convergence rate: the larger the step size, the faster the convergence. On the other hand, η_k is constrained by the condition in (9), which, in turn, depends on the Hessian approximation matrix $\mathbf{B}_k$. Thus, the central goal of our line search scheme and the Hessian approximation update is to make our step size η_k as large as possible.

3.1. Backtracking line search

In this section, we describe a backtracking line search scheme for selecting the step size η_k and the iterate $\hat{\mathbf{x}}_k$ in the first stage of QNPE. For simplicity, we denote $\nabla f(\mathbf{x}_k)$ by $\mathbf{g}$ and drop the subscript k in $\mathbf{x}_k$ and $\mathbf{B}_k$. Recall that at the k-th iteration, our goal is to find a pair $(\eta_+, \hat{\mathbf{x}}_+)$ such that

$$\|\hat{\mathbf{x}}_+ - \mathbf{x} + \eta_+(\mathbf{g} + \mathbf{B}(\hat{\mathbf{x}}_+ - \mathbf{x}))\| \leq \alpha_1 \|\hat{\mathbf{x}}_+ - \mathbf{x}\|, \tag{11}$$

$$\eta_+ \|\nabla f(\hat{\mathbf{x}}_+) - \mathbf{g} - \mathbf{B}(\hat{\mathbf{x}}_+ - \mathbf{x})\| \leq \alpha_2 \|\hat{\mathbf{x}}_+ - \mathbf{x}\|. \tag{12}$$

As mentioned in the previous section, the condition in (11) can be satisfied if we solve the linear system $(\mathbf{I} + \eta_+\mathbf{B})(\hat{\mathbf{x}}_+ - \mathbf{x}) = -\eta_+\mathbf{g}$ to a desired accuracy. To formalize, we let

$$\mathbf{s}_+ = \mathsf{LinearSolver}(\mathbf{I} + \eta_+\mathbf{B}, -\eta_+\mathbf{g}; \alpha_1) \quad \text{and} \quad \hat{\mathbf{x}}_+ = \mathbf{x} + \mathbf{s}_+, \tag{13}$$

Subroutine 1 Backtracking line search

```
1:  Input: iterate $\mathbf{x} \in \mathbb{R}^d$, gradient $\mathbf{g} \in \mathbb{R}^d$, Hessian approximation $\mathbf{B} \in \mathbb{S}^d_+$, initial trial step size $\sigma > 0$
2:  Parameters: line search parameters $\beta \in (0,1)$, $\alpha_1 \geq 0$ and $\alpha_2 > 0$ such that $\alpha_1 + \alpha_2 < 1$
3:  Set $\eta_+ \leftarrow \sigma$, $\mathbf{s}_+ \leftarrow \mathsf{LinearSolver}(\mathbf{I} + \eta_+\mathbf{B}, -\eta_+\mathbf{g}; \alpha_1)$ and $\hat{\mathbf{x}}_+ \leftarrow \mathbf{x} + \mathbf{s}_+$
4:  while $\eta_+\|\nabla f(\hat{\mathbf{x}}_+) - \mathbf{g} - \mathbf{B}(\hat{\mathbf{x}}_+ - \mathbf{x})\|_2 > \alpha_2\|\hat{\mathbf{x}}_+ - \mathbf{x}\|_2$ do
5:      Set $\tilde{\mathbf{x}} \leftarrow \hat{\mathbf{x}}_+$ and $\eta_+ \leftarrow \beta\eta_+$
6:      Compute $\mathbf{s}_+ \leftarrow \mathsf{LinearSolver}(\mathbf{I} + \eta_+\mathbf{B}, -\eta_+\mathbf{g}; \alpha_1)$ and $\hat{\mathbf{x}}_+ \leftarrow \mathbf{x} + \mathbf{s}_+$
7:  end while
8:  if $\eta_+ = \sigma$ then
9:      Return $\eta_+$ and $\hat{\mathbf{x}}_+$
10: else
11:     Return $\eta_+$, $\hat{\mathbf{x}}_+$ and $\tilde{\mathbf{x}}$
12: end if
```

where the oracle LinearSolver is defined as follows.

Definition 2 *The oracle* $\mathsf{LinearSolver}(\mathbf{A}, \mathbf{b}; \alpha)$ *takes a matrix* $\mathbf{A} \in \mathbb{S}^d_+$*, a vector* $\mathbf{b} \in \mathbb{R}^d$ *and* $\alpha \in (0,1)$ *as input, and returns an approximate solution* $\mathbf{s}_+$ *satisfying* $\|\mathbf{A}\mathbf{s}_+ - \mathbf{b}\| \leq \alpha\|\mathbf{s}_+\|$.

By Definition 2, the pair $(\eta_+, \hat{\mathbf{x}}_+)$ is guaranteed to satisfy (11) when $\hat{\mathbf{x}}_+$ is computed based on (13). To implement the oracle $\mathsf{LinearSolver}(\mathbf{A}, \mathbf{b}; \alpha)$, the most direct way is to compute the exact solution $\mathbf{s}_+ = \mathbf{A}^{-1}\mathbf{b}$. In Appendix C.1, we will describe a more efficient implementation via the conjugate residual method (Saad, 2003), which only requires computing matrix-vector products.

Now we are ready to describe our backtracking line search scheme in Subroutine 1 assuming access to the LinearSolver oracle. Specifically, given a user-defined parameter $\beta \in (0,1)$ and initial trial step size $\sigma > 0$, we try the step sizes in $\{\sigma\beta^i : i \geq 0\}$ in decreasing order and compute $\hat{\mathbf{x}}_+$ according to (13), until we find one pair $(\eta_+, \hat{\mathbf{x}}_+)$ that satisfies (12). Note that by standard arguments, the line search scheme is guaranteed to terminate in a finite number of steps. Since (11) already holds true by following the update rule in (13), the line search scheme will return a pair $(\eta_+, \hat{\mathbf{x}}_+)$ satisfying both conditions in (11) and (12) . Regarding the output, we distinguish two cases. If we pass the test in (12) on our first attempt, we accept the initial step size σ and the corresponding iterate $\hat{\mathbf{x}}_+$ (cf. Line 9). Otherwise, if σ fails the test and we go through the backtracking procedure, along with the pair $(\eta_+, \hat{\mathbf{x}}_+)$, we also return an auxiliary iterate $\tilde{\mathbf{x}}$, which is the last rejected point we compute from (13) using the step size η_+/β (cf. Line 11). As we shall see in Lemma 3, the iterate $\tilde{\mathbf{x}}$ is used to construct a lower bound on η_+, which will guide our update of the Hessian approximation matrix.

For ease of notation, let $\mathcal{B}$ be the set of iteration indices where the line search scheme backtracks, i.e., $\mathcal{B} \triangleq \{k : \eta_k < \sigma_k\}$. For these iterations in $\mathcal{B}$, the next lemma provides a lower bound on the step size η_k returned by our line search scheme, which will be the key to our convergence analysis and the update of the Hessian approximation matrices. The proof can be found in Appendix A.2.

Lemma 3 *For* $k \notin \mathcal{B}$ *we have* $\eta_k = \sigma_k$*, while for* $k \in \mathcal{B}$ *we have*

$$\eta_k > \frac{\alpha_2\beta\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|}{\|\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\|} \quad \textit{and} \quad \|\tilde{\mathbf{x}}_k - \mathbf{x}_k\| \leq \frac{1+\alpha_1}{\beta(1-\alpha_1)}\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|. \tag{14}$$

In Lemma 3, we lower bound the step size η_k in terms of the ratio between $\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|$ and the approximation error $\|\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\|$. Hence, a better Hessian approximation matrix $\mathbf{B}_k$ leads to a larger step size, which in turn implies faster convergence. Also, we note that the lower bound depends on the auxiliary iterate $\tilde{\mathbf{x}}_k$ that is not accepted as the actual iterate. As such, we will use the second inequality in (14) to relate $\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|$ with $\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|$. Finally, we remark that to fully characterize the computational cost of our method, we need to upper bound the total number of line search steps, each of which requires a call to LinearSolver and a call to the gradient oracle. This will be discussed later in Section 4.1.

3.2. Hessian Approximation Update via Online Learning

In this section, we focus on the update rule for the Hessian approximation matrix $\mathbf{B}_k$. Our goal is to develop a policy that leads to an explicit superlinear convergence rate for our proposed QNPE method. As mentioned earlier, our new policy differs greatly from the traditional quasi-Newton updates and is solely guided by the convergence analysis of our method.

Our starting point is Proposition 1, which characterizes the convergence rate of QNPE in terms of the step size η_k. It implies that if we can show $\eta_k \to \infty$, then a superlinear convergence rate follows immediately. Specifically, by repeatedly applying the result of Proposition 1, we obtain

$$\frac{\|\mathbf{x}_N - \mathbf{x}^*\|^2}{\|\mathbf{x}_0 - \mathbf{x}^*\|^2} \leq \prod_{k=0}^{N-1} (1 + 2\eta_k \mu)^{-1} \leq \left(1 + 2\mu \sqrt{\frac{N}{\sum_{k=0}^{N-1} 1/\eta_k^2}}\right)^{-N}, \tag{15}$$

where the last inequality follows from Jensen's inequality applied to the convex function $t \mapsto \log(1 + \frac{1}{t})$. Hence, if we upper bound $\sum_{k=0}^{N-1} 1/\eta_k^2$ by a constant independent of N, it implies a global superlinear convergence rate of $\mathcal{O}(1/\sqrt{N})^N$. Moreover, Lemma 3 gives us the tool to control the step sizes and establish an upper bound on $\sum_{k=0}^{N-1} 1/\eta_k^2$, as shown in the following lemma. The proof is given in Appendix A.3.

Lemma 4 *Let $\{\eta_k\}_{k=0}^{N-1}$ be the step sizes in Algorithm 1 using the line search in Subroutine 1. Then,*

$$\sum_{k=0}^{N-1} \frac{1}{\eta_k^2} \leq \frac{1}{(1-\beta^2)\sigma_0^2} + \frac{1}{(1-\beta^2)\alpha_2^2\beta^2} \sum_{k \in \mathcal{B}} \frac{\|\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k\|^2}{\|\mathbf{s}_k\|^2}, \tag{16}$$

where $\mathbf{y}_k \triangleq \nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k)$ and $\mathbf{s}_k \triangleq \tilde{\mathbf{x}}_k - \mathbf{x}_k$.

Our key idea is to interpret the right-hand side of (16) as the cumulative loss incurred by our choice of $\mathbf{B}_k$, and to update the Hessian approximation matrix by an online learning algorithm. More formally, define the loss function at iteration k as

$$\ell_k(\mathbf{B}) \triangleq \begin{cases} 0, & \text{if } k \notin \mathcal{B}, \\ \frac{\|\mathbf{y}_k - \mathbf{B}\mathbf{s}_k\|^2}{2\|\mathbf{s}_k\|^2}, & \text{otherwise.} \end{cases} \tag{17}$$

Then the online learning protocol works as follows: (i) At the k-th iteration, we choose $\mathbf{B}_k \in \mathcal{Z}'$, where $\mathcal{Z}' \triangleq \{\mathbf{B} \in \mathbb{S}_+^d : \frac{\mu}{2}\mathbf{I} \preceq \mathbf{B} \preceq (L_1 + \frac{\mu}{2})\mathbf{I}\}$; (ii) We receive the loss function $\ell_k(\mathbf{B})$ defined in (17); (iii) We update our Hessian approximation to $\mathbf{B}_{k+1}$. Hence, minimizing the sum in (16) is equivalent to minimizing the cumulative loss $\sum_{k=0}^{N-1} \ell_k(\mathbf{B}_k)$, which is exactly what online learning algorithms are designed for. In particular, we will show in Lemma 12 that the cumulative loss $\sum_{k=0}^{N-1} \ell_k(\mathbf{B}_k)$ incurred by our online learning algorithm is comparable to $\sum_{k=0}^{N-1} \ell_k(\mathbf{H}^*)$, where $\mathbf{H}^* \triangleq \nabla^2 f(\mathbf{x}^*)$ is the exact Hessian at the optimal solution $\mathbf{x}^*$.

Remark 5 *By Assumption 1, we know that $\mu\mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq L_1\mathbf{I}$. Thus, it is natural to restrict $\mathbf{B}_k$ to the set $\mathcal{Z} \triangleq \{\mathbf{B} \in \mathbb{S}_+^d : \mu\mathbf{I} \preceq \mathbf{B} \preceq L_1\mathbf{I}\}$. On the other hand, this constraint is by no means mandatory, and looser bounds on the eigenvalues of $\mathbf{B}_k$ would also suffice for our analysis. Hence, we exploit this flexibility and allow our algorithm to pick $\mathbf{B}_k$ from a larger set $\mathcal{Z}'$, as it is easier to enforce such a constraint. We discuss this point in detail in Section 3.2.2.*

Remark 6 *Since we have $\ell_k(\mathbf{B}) = 0$ when $k \notin \mathcal{B}$, we can simply keep $\mathbf{B}_{k+1}$ unchanged for these iterations (cf. Line 8 in Algorithm 1). With a slight abuse of notation, in the following, we relabel the indices in $\mathcal{B}$ by $t = 0, 1, \ldots, T-1$ with $T \leq N$.*

Now that we formulated the Hessian approximation update as an online learning problem, one can update $\mathbf{B}_k$ by an online learning method, such as the projected online gradient descent (Zinkevich, 2003). This approach would indeed serve our purpose and lead to an explicit superlinear convergence rate. However, in our setting, implementing any projection-based online learning algorithm could be computationally expensive: the Euclidean projection onto the set $\mathcal{Z}'$ requires performing a full $d \times d$ matrix eigendecomposition, which typically incurs a complexity of $\mathcal{O}(d^3)$; please check Appendix D.1 for more discussions. In the following, we instead build upon a projection-free online learning algorithm proposed by Mhammedi (2022).

3.2.1. ONLINE LEARNING WITH AN APPROXIMATE SEPARATION ORACLE

To better illustrate our key idea, we take a step back and consider a general online learning problem. For T consecutive rounds $t = 0, 1, \ldots, T-1$, a learner chooses an action $\mathbf{x}_t \in \mathbb{R}^n$ from an action set and then observes a loss function $\ell_t : \mathbb{R}^n \to \mathbb{R}$. The goal is to minimize the regret defined by $\mathrm{Reg}_T(\mathbf{x}) \triangleq \sum_{t=0}^{T-1} \ell_t(\mathbf{x}_t) - \sum_{t=0}^{T-1} \ell_t(\mathbf{x})$, which is the difference between the cumulative loss of the learner and that of a fixed competitor $\mathbf{x}$. This is a standard online learning problem, but with a slight modification: we restrict the competitor $\mathbf{x}$ to be in a given competitor set $\mathcal{C}$, while allow the learner to choose the action $\mathbf{x}_t$ from a larger set $(1+\delta)\mathcal{C} \triangleq \{(1+\delta)\mathbf{x} : \mathbf{x} \in \mathcal{C}\}$ for some given $\delta > 0$. As mentioned in Remark 5, this setup is more suitable for our Hessian approximation update framework, where the constraint on $\mathbf{B}_t$ is more flexible (note that $\mathbf{x}_t$ and $\mathbf{x}$ correspond to $\mathbf{B}_t$ and $\mathbf{H}^*$, respectively). Finally, without loss of generality, we can assume that $0 \in \mathcal{C}$. We also assume that the convex set $\mathcal{C}$ is bounded and contained in the Euclidean ball $\mathcal{B}_R(0)$ for some $R > 0$.

To solve the above online learning problem, most existing algorithms require access to an oracle that computes the Euclidean projection on the action set. However, computing the projection is computationally costly in our setting (see Appendix D.1). Unlike these projection-based methods, here we rely on an approximate separation oracle defined below. As we discuss in Appendix C.2, the following SEP oracle can be implemented much more efficiently than the Euclidean projection.

Definition 7 *The oracle $\mathsf{SEP}(\mathbf{w}; \delta)$ takes $\mathbf{w} \in \mathcal{B}_R(0)$ and $\delta > 0$ as input and returns a scalar $\gamma > 0$ and a vector $\mathbf{s} \in \mathbb{R}^n$ with one of the following possible outcomes:*

- *Case I: $\gamma \leq 1$ which implies that $\mathbf{w} \in (1+\delta)\mathcal{C}$;*
- *Case II: $\gamma > 1$ which implies that $\mathbf{w}/\gamma \in (1+\delta)\mathcal{C}$ and $\langle \mathbf{s}, \mathbf{w} - \mathbf{x} \rangle \geq \gamma - 1 \quad \forall \mathbf{x} \in \mathcal{C}$.*

In summary, the oracle $\mathsf{SEP}(\mathbf{w}; \delta)$ has two possible outcomes: it either certifies that $\mathbf{w}$ is approximately feasible, i.e., $\mathbf{w} \in (1+\delta)\mathcal{C}$, or it produces a scaled version of $\mathbf{w}$ that is in $(1+\delta)\mathcal{C}$ and gives a strict separating hyperplane between $\mathbf{w}$ and the set $\mathcal{C}$.

Remark 8 *There are two main differences between Algorithm 1 in (Mhammedi, 2022) and our presentation here. First, Mhammedi (2022) considered a standard online learning setup where the action $\mathbf{x}_t$ must be in the competitor set $\mathcal{C}$, while in our setting $\mathbf{x}_t$ can be chosen from a larger set $(1+\delta)\mathcal{C}$. Second, their algorithm relied on an oracle that approximates the gauge function $\gamma_{\mathcal{C}}(\mathbf{w}) \triangleq \inf\{\lambda \geq 0 : \mathbf{w} \in \lambda\mathcal{C}\}$ and its subgradient, which is further explicitly constructed using a membership oracle. Our oracle in Definition 7 is different but related, in the sense that its output γ and $\mathbf{s}$ may also be regarded as an approximation of the gauge function and its subgradient. Moreover, we focus on the specific set used in our Hessian approximation update, and offer a more accustomed regret analysis and efficient construction of the oracle.*

The key idea here is to introduce an auxiliary online learning problem on the larger set $\mathcal{B}_R(0)$ with surrogate loss functions $\tilde{\ell}_t(\mathbf{w}) = \langle \tilde{\mathbf{g}}_t, \mathbf{w} \rangle$ for $0 \leq t \leq T-1$, where $\tilde{\mathbf{g}}_t$ is the surrogate gradient to be defined later. On a high level, we will run online projected gradient descent on this auxiliary problem to update the iterates $\{\mathbf{w}_t\}_{t\geq 0}$ (note that the projection on $\mathcal{B}_R(0)$ is easy to compute), and then produce the actions $\{\mathbf{x}_t\}_{t\geq 0}$ for the original problem by calling $\mathsf{SEP}(\mathbf{w}_t;\delta)$ in Definition 7. Specifically, given $\mathbf{w}_t$ at round t, we let $\gamma_t > 0$ and $\mathbf{s}_t \in \mathbb{R}^n$ be the output of $\mathsf{SEP}(\mathbf{w}_t;\delta)$. If $\gamma_t \leq 1$, we are in **Case I**, where we set $\mathbf{x}_t = \mathbf{w}_t$, compute $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$, and define the surrogate gradient by $\tilde{\mathbf{g}}_t = \mathbf{g}_t$. Otherwise, if $\gamma_t > 1$, we are in **Case II**, where we set $\mathbf{x}_t = \mathbf{w}_t/\gamma_t$, compute $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$, and define the surrogate gradient by $\tilde{\mathbf{g}}_t = \mathbf{g}_t + \max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t\rangle\}\mathbf{s}_t$. Note that Definition 7 guarantees $\mathbf{x}_t \in (1+\delta)\mathcal{C}$ in both cases. Finally, we update $\mathbf{w}_{t+1}$ using the standard online projected gradient descent with respect to the surrogate loss $\tilde{\ell}_t(\mathbf{w})$ and the set $\mathcal{B}_R(0)$:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{B}_R(0)}(\mathbf{w}_t - \rho\tilde{\mathbf{g}}_t) = \frac{R}{\max\{\|\mathbf{w}_t - \rho\tilde{\mathbf{g}}_t\|_2, R\}}(\mathbf{w}_t - \rho\tilde{\mathbf{g}}_t),$$

where ρ is the step size. To give some intuition, the surrogate loss functions $\{\tilde{\ell}_t(\mathbf{w})\}_{t=1}^T$ are constructed in such a way that the immediate regret $\tilde{\ell}_t(\mathbf{w}_t) - \tilde{\ell}_t(\mathbf{x})$ serves as an upper bound on $\ell_t(\mathbf{x}_t) - \ell_t(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{C}$. Therefore, we can upper bound the regret of the original problem by that of the auxiliary problem, which can be further bounded using the standard analysis for online projected gradient descent. The full algorithm is described in Algorithm 2 in Appendix B.2.

3.2.2. Projection-free Hessian Approximation Update

Now we are ready to describe our projection-free online learning algorithm for updating $\mathbf{B}_k$, which is a special case of the algorithm described in the previous section. Recall that in our online learning problem in Section 3.2, the competitor set is $\mathcal{Z} = \{\mathbf{B} \in \mathbb{S}^d_+ : \mu\mathbf{I} \preceq \mathbf{B} \preceq L_1\mathbf{I}\}$. Since the discussed projection-free scheme requires the competitor set $\mathcal{C}$ to contain the origin, we first translate and rescale $\mathbf{B}$ via the transform $\hat{\mathbf{B}} \triangleq \frac{2}{L_1-\mu}(\mathbf{B} - \frac{L_1+\mu}{2}\mathbf{I})$ and define $\mathcal{C} \triangleq \{\hat{\mathbf{B}} \in \mathbb{S}^d : -\mathbf{I} \preceq \hat{\mathbf{B}} \preceq \mathbf{I}\} = \{\hat{\mathbf{B}} \in \mathbb{S}^d : \|\hat{\mathbf{B}}\|_{\mathrm{op}} \leq 1\}$, which satisfies $0 \in \mathcal{C}$ and $\mathcal{C} \subset \mathcal{B}_{\sqrt{d}}(0) = \{\mathbf{W} \in \mathbb{S}^d : \|\mathbf{W}\|_F \leq \sqrt{d}\}$. It can be verified that $\mathbf{B} \in \mathcal{Z} \iff \hat{\mathbf{B}} \in \mathcal{C}$, and also $\mathbf{B} \in \mathcal{Z}' \iff \hat{\mathbf{B}} \in (1+\delta)\mathcal{C}$ with $\delta = \mu/(L_1-\mu)$.

The only remaining question is how we can build the SEP oracle in Definition 7 for our specific set $\mathcal{C}$. To begin with, we observe that this is closely related to computing the extreme eigenvalues and the associated eigenvectors of a given matrix $\mathbf{W}$. In fact, let $\lambda_{\max}$ and $\mathbf{v}_{\max} \in \mathbb{R}^d$ be the largest magnitude eigenvalue of $\mathbf{W}$ and its associated unit eigenvector, respectively. Since $\|\mathbf{W}\|_{\mathrm{op}} = |\lambda_{\max}|$, it is easy to see that: (i) If $|\lambda_{\max}| \leq 1$, then $\mathbf{W} \in \mathcal{C}$; (ii) Otherwise, if $|\lambda_{\max}| > 1$, then we can let $\gamma = |\lambda_{\max}|$, which satisfies $\mathbf{W}/\gamma \in \mathcal{C}$, and $\mathbf{S} = \mathrm{sign}(\lambda_{\max})\mathbf{v}_{\max}\mathbf{v}_{\max}^\top \in \mathbb{S}^d$, which defines

Subroutine 2 Online Learning Guided Hessian Approximation Update

```
Input: Initial matrix $\mathbf{B}_0 \in \mathbb{S}^d$ s.t. $\mu\mathbf{I} \preceq \mathbf{B}_0 \preceq L_1\mathbf{I}$, step size $\rho > 0$, $\delta > 0$, $\{q_t\}_{t=1}^{T-1}$
Initialize: set $\mathbf{W}_0 \leftarrow \frac{2}{L_1-\mu}(\mathbf{B}_0 - \frac{L_1+\mu}{2}\mathbf{I})$, $\mathbf{G}_0 \leftarrow \frac{2}{L_1+\mu}\nabla\ell_0(\mathbf{B}_0)$ and $\tilde{\mathbf{G}}_0 \leftarrow \mathbf{G}_0$
Update $\mathbf{W}_1 \leftarrow \frac{\sqrt{d}}{\max\{\sqrt{d}, \|\mathbf{W}_0 - \rho\tilde{\mathbf{G}}_0\|_F\}}(\mathbf{W}_0 - \rho\tilde{\mathbf{G}}_0)$
for $t = 1, \ldots, T-1$ do
    Query the oracle $(\gamma_t, \mathbf{S}_t) \leftarrow \mathsf{ExtEvec}(\mathbf{W}_t; \delta, q_t)$
    if $\gamma_t \leq 1$ then    # Case I
        Set $\hat{\mathbf{B}}_t \leftarrow \mathbf{W}_t$ and $\mathbf{B}_t \leftarrow \frac{L_1-\mu}{2}\hat{\mathbf{B}}_t + \frac{L_1+\mu}{2}\mathbf{I}$
        Set $\mathbf{G}_t \leftarrow \frac{2}{L_1-\mu}\nabla\ell_t(\mathbf{B}_t)$ and $\tilde{\mathbf{G}}_t \leftarrow \mathbf{G}_t$
    else    # Case II
        Set $\hat{\mathbf{B}}_t \leftarrow \mathbf{W}_t/\gamma_t$ and $\mathbf{B}_t \leftarrow \frac{L_1-\mu}{2}\hat{\mathbf{B}}_t + \frac{L_1+\mu}{2}\mathbf{I}$
        Set $\mathbf{G}_t \leftarrow \frac{2}{L_1-\mu}\nabla\ell_t(\mathbf{B}_t)$ and $\tilde{\mathbf{G}}_t \leftarrow \mathbf{G}_t + \max\{0, -\langle\mathbf{G}_t, \mathbf{B}_t\rangle\}\mathbf{S}_t$
    end if
    Update $\mathbf{W}_{t+1} \leftarrow \frac{\sqrt{d}}{\max\{\sqrt{d}, \|\mathbf{W}_t - \rho\tilde{\mathbf{G}}_t\|_F\}}(\mathbf{W}_t - \rho\tilde{\mathbf{G}}_t)$    # Euclidean projection onto $\mathcal{B}_{\sqrt{d}}(0)$
end for
```

a separating hyperplane between $\mathbf{W}$ and $\mathcal{C}$. Indeed, note that we have $\langle\mathbf{S}, \mathbf{W}\rangle = |\lambda_{\max}| = \gamma$ and $\langle\mathbf{S}, \hat{\mathbf{B}}\rangle \leq |\mathbf{v}_{\max}^\top\hat{\mathbf{B}}\mathbf{v}_{\max}| \leq 1$ for any $\hat{\mathbf{B}} \in \mathcal{C}$, which implies $\langle\mathbf{S}, \mathbf{W} - \hat{\mathbf{B}}\rangle \geq \gamma - 1$. Hence, we can build the separation oracle in Definition 7 if we compute $\lambda_{\max}$ and $\mathbf{v}_{\max}$ for the given matrix $\mathbf{W}$.

However, the exact computation of $\lambda_{\max}$ and $\mathbf{v}_{\max}$ could be costly. Thus, we propose to compute the extreme eigenvalues and the corresponding eigenvectors inexactly by the randomized Lanczos method (Kuczyński and Woźniakowski, 1992), which leads to the randomized oracle ExtEvec defined below. We defer the specific implementation details of ExtEvec to Section C.2.

Definition 9 *The oracle* ExtEvec$(\mathbf{W}; \delta, q)$ *takes* $\mathbf{W} \in \mathbb{S}^d$, $\delta > 0$, *and* $q \in (0, 1)$ *as input and returns a scalar* $\gamma > 0$ *and a matrix* $\mathbf{S} \in \mathbb{S}^d$ *with one of the following possible outcomes:*

- *Case I:* $\gamma \leq 1$, *which implies that, with probability at least* $1 - q$, $\|\mathbf{W}\|_{\text{op}} \leq 1 + \delta$*;*
- *Case II:* $\gamma > 1$, *which implies that, with probability at least* $1 - q$, $\|\mathbf{W}/\gamma\|_{\text{op}} \leq 1 + \delta$, $\|\mathbf{S}\|_F = 1$ *and* $\langle\mathbf{S}, \mathbf{W} - \hat{\mathbf{B}}\rangle \geq \gamma - 1$ *for any* $\hat{\mathbf{B}}$ *such that* $\|\hat{\mathbf{B}}\|_{\text{op}} \leq 1$.

Note that ExtEvec is an approximate separation oracle for the set $\mathcal{C}$ in the sense of Definition 7 (with success probability at least $1 - q$), and it also guarantees that $\|\mathbf{S}\|_F = 1$ in Case II. Equipped with this oracle, we describe the complete Hessian approximation update in Subroutine 2.

4. Complexity Analysis of QNPE

By now, we have fully described our QNPE method in Algorithm 1, where we select the η_k by Subroutine 1 and update the Hessian approximation matrix $\mathbf{B}_k$ by Subroutine 2. In the following, we shall establish the convergence rate and characterize the computational cost of QNPE.

Next, we state our main convergence result. Our results hold for any $\alpha_1, \alpha_2 \in (0, \frac{1}{2})$ and $\beta \in (0, 1)$, but to simplify our expressions we report the results for $\alpha_1 = \alpha_2 = \frac{1}{4}$ and $\beta = \frac{1}{2}$.

Theorem 10 (Main Theorem) *Let* $\{\mathbf{x}_k\}_{k\geq 0}$ *be the iterates generated by Algorithm 1 using the line search scheme in Subroutine 1, where* $\alpha_1 = \alpha_2 = \frac{1}{4}$, $\beta = \frac{1}{2}$, *and* $\sigma_0 \geq \alpha_2\beta/L_1$, *and the Hessian approximation update in Subroutine 2, where* $\rho = \frac{1}{18}$, $\delta = \min\{\frac{\mu}{L_1-\mu}, 1\}$, *and* $q_t = {}^{p}/_{2.5(t+1)\log^2(t+1)}$ *for* $t \geq 1$. *Then with probability at least* $1 - p$, *the following statements hold:*

(a) *(Linear convergence) For any $k \geq 0$, we have* $\frac{\|\mathbf{x}_{k+1}-\mathbf{x}^*\|^2}{\|\mathbf{x}_k-\mathbf{x}^*\|^2} \leq \left(1+\frac{\mu}{4L_1}\right)^{-1}$.

(b) *(Superlinear convergence) We have* $\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1}-\mathbf{x}^*\|^2}{\|\mathbf{x}_k-\mathbf{x}^*\|^2} = 0$. *Furthermore, for any $k \geq 0$,*

$$\frac{\|\mathbf{x}_k-\mathbf{x}^*\|^2}{\|\mathbf{x}_0-\mathbf{x}^*\|^2} \leq \left(1+\frac{\sqrt{3}}{8}\mu\sqrt{\frac{k}{L_1^2+36\|\mathbf{B}_0-\nabla^2 f(\mathbf{x}^*)\|_F^2+\left(27+\frac{16L_1}{\mu}\right)L_2^2\|\mathbf{x}_0-\mathbf{x}^*\|^2}}\right)^{-k}.$$

Proof sketch. By using a simple union bound, we can show that the ExtEvec oracle in Subroutine 2 is successful in all rounds with probability at least $1-p$. Thus, throughout the proof, we assume that every call of ExtEvec is successful. We first prove the linear convergence rate in (a). As we discussed in Section 3.2.2, Subroutine 2 ensures that $\frac{\mu}{2}\mathbf{I} \preceq \mathbf{B}_k \preceq L_1+\frac{\mu}{2}\mathbf{I}$ for any $k \geq 0$. Combining this with Lemma 3, we obtain the following universal lower bound on the step size η_k.

Lemma 11 *For any $k \geq 0$, we have $\eta_k \geq 1/(8L_1)$.*

In light of Lemma 11, the linear convergence result in (a) now follows directly from Proposition 1.

Next, we prove the superlinear convergence rate in (b) by considering the following steps.

Step 1: Using regret analysis, we bound the cumulative loss $\sum_{t=0}^{T-1}\ell_t(\mathbf{B}_t)$ incurred by our online learning algorithm in Subroutine 2. In particular, by exploiting the smooth property of the loss function ℓ_t, we prove a small-loss bound in the following lemma, where the cumulative loss of the learner is bounded by that of a fixed action in the competitor set (Srebro et al., 2010).

Lemma 12 *For any $\mathbf{H} \in \mathcal{Z}$, we have* $\sum_{t=0}^{T-1}\ell_t(\mathbf{B}_t) \leq 18\|\mathbf{B}_0-\mathbf{H}\|_F^2 + 2\sum_{t=0}^{T-1}\ell_t(\mathbf{H})$.

Note that in Lemma 12, we have the freedom to choose any competitor $\mathbf{H}$ in the set $\mathcal{Z}$. To further obtain an explicit bound, a natural choice would be $\mathbf{H}^* \triangleq \nabla^2 f(\mathbf{x}^*)$, which leads to our next step.

Step 2: We upper bound the cumulative loss $\sum_{t=0}^{T-1}\ell_t(\mathbf{H}^*)$ in the following lemma. The proof relies crucially on Assumption 2 as well as the linear convergence result we proved in (a).

Lemma 13 *We have* $\sum_{t=0}^{T-1}\ell_t(\mathbf{H}^*) \leq \left(\frac{27}{4}+\frac{4L_1}{\mu}\right)L_2^2\|\mathbf{x}_0-\mathbf{x}^*\|^2$.

Step 3: Combining Lemma 13 and Lemma 12, we obtain a constant upper bound on the cumulative loss $\sum_{t=0}^{T-1}\ell_t(\mathbf{B}_t)$. By Lemma 4, this further implies an upper bound on $\sum_{k=0}^{N-1}1/\eta_k^2$, which leads to the superlinear convergence result in (b) by Proposition 1 and the observation in (15). ■

Discussions. To begin with, Part (a) of Theorem 10 guarantees that QNPE converges linearly and is at least as fast as gradient descent. Moreover, in Part (b) we prove Q-superlinear convergence of QNPE, where the explicit global superlinear rate is faster than the linear rate for sufficiently large k. Specifically, if we define $N_{\text{tr}} \triangleq \frac{4}{3}+\frac{48}{L_1^2}\|\mathbf{B}_0-\nabla^2 f(\mathbf{x}^*)\|_F^2+\left(\frac{36}{L_1^2}+\frac{64}{3\mu L_1}\right)L_2^2\|\mathbf{x}_0-\mathbf{x}^*\|^2$, then the superlinear rate can be written as $(1+\frac{\mu}{4L_1}\sqrt{\frac{k}{N_{\text{tr}}}})^{-k}$, which is superior to the linear rate when $k \geq N_{\text{tr}}$. Moreover, we can also derive an explicit complexity bound from Theorem 10. Let N_ϵ denote the number of iterations required by QNPE to achieve ϵ-accurate solution, i.e., $\|\mathbf{x}_k-\mathbf{x}^*\|^2 \leq \epsilon$. As we show in Appendix D.2, if the error tolerance ϵ is in the regime where $\epsilon > \exp(-\frac{\mu}{L_1}N_{\text{tr}})$, the linear rate in Part (a) is faster and we have $N_\epsilon = \mathcal{O}(\frac{L_1}{\mu}\log\frac{1}{\epsilon})$. Otherwise, if $\epsilon < \exp(-\frac{\mu}{L_1}N_{\text{tr}})$, the superlinear rate in Part (b) excels and we have $N_\epsilon = \mathcal{O}\Big(\Big[\log\Big(1+\frac{\mu}{L_1}\Big(\frac{L_1}{N_{\text{tr}}\mu}\log\frac{1}{\epsilon}\Big)^{1/3}\Big)\Big]^{-1}\log\frac{1}{\epsilon}\Big)$.

A couple of additional remarks about Theorem 10 follow. First, the expression $\|\mathbf{B}_0 - \nabla^2 f(\mathbf{x}^*)\|_F^2$ is bounded above by $L_1^2 d$ in the worst-case, showing that at worst N_{tr} scales linearly with the dimension d. On the other hand, N_{tr} could be much smaller if the initial Hessian approximation matrix $\mathbf{B}_0$ is close to $\nabla^2 f(\mathbf{x}^*)$. Second, Theorem 10 provides a global result, as both bounds hold for any initial point $\mathbf{x}_0$ and any initial Hessian approximation $\mathbf{B}_0$. On the contrary, the existing non-asymptotic results on quasi-Newton methods in (Rodomanov and Nesterov, 2021c; Jin and Mokhtari, 2022) require special initialization for $\mathbf{B}_0$ and closeness of $\mathbf{x}_0$ to the optimal solution $\mathbf{x}^*$.

4.1. Characterizing the Computational Cost

As for most optimization algorithms, we measure the computational cost of our QNPE method in two aspects: the number of gradient evaluations and the number of matrix-vector product evaluations. In particular, each backtracking step of the line search scheme in Subroutine 1 requires one call to the gradient oracle, while the implementation of LinearSolver in Definition 2 and ExtEvec in Definition 9 requires multiple matrix-vector products. Due to space limitations, we defer the details to Appendix C and summarize the complexity results in the following theorem.

Theorem 14 *Let N_ϵ denote the minimum number of iterations required by Algorithm 1 to find an ϵ-accurate solution according to Theorem 10. Then, with probability at least $1-p$:*

(a) The total number of gradient evaluations is bounded by $3N_\epsilon + \log_{1/\beta}(4\sigma_0 L_1)$.

(b) The total number of matrix-vector products in ExtEvec *and* LinearSolver *are bounded by $\mathcal{O}\left(N_\epsilon \sqrt{\frac{L_1}{\mu}} \log\left(\frac{dN_\epsilon^2}{p^2}\right)\right)$ and $\mathcal{O}\left(N_\epsilon \sqrt{\frac{L_1}{\mu}} \log\left(\frac{L_1\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\mu\epsilon}\right)\right)$, respectively.*

As a direct corollary, on average QNPE requires at most 3 gradient evaluations per iteration if we set $\sigma_0 = 1/(4L_1)$. Moreover, by summing the complexity of both ExtEvec and LinearSolver, we can bound the total number of matrix-vector products by $\mathcal{O}(N_\epsilon \sqrt{\frac{L_1}{\mu}} \log \frac{L_1 N_\epsilon^2 d}{\mu\epsilon})$.

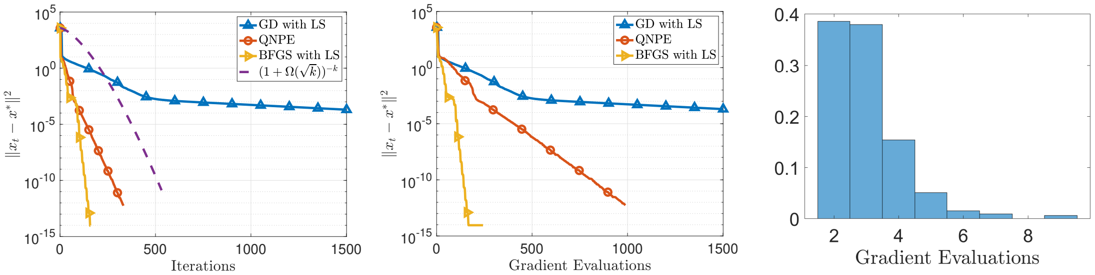
5. Numerical Experiments

To verify our theoretical findings, we consider a regularized logistic regression problem on a synthetic dataset $\{(\mathbf{a}_i, y_i)\}_{i=1}^n$, where $\mathbf{a}_i \in \mathbb{R}^d$ is the i-th feature vector and $y_i \in \{+1, -1\}$ is the i-th binary label (details on the dataset can be found in Appendix E). It can be formulated as the following optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} \quad f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^n \log(1 + e^{-y_i\langle \mathbf{a}_i, \mathbf{x}\rangle}) + \frac{\mu}{2}\|\mathbf{x}\|^2,$$

where μ is the regularization parameter. In our experiment, we set $d = 150$, $n = 2000$ and $\mu = 0.005$, with the condition number L_1/μ estimated to be 7600.

We implemented our proposed method QNPE following Algorithm 1, where we select the step size η_k by Subroutine 1 and update the Hessian approximation matrix $\mathbf{B}_k$ by Subroutine 2. Moreover, the LinearSolver oracle is implemented using the conjugate residual method (see Subroutine 3), while the ExtEvec oracle is implemented using MATLAB's eig function (we can afford full eigendecomposition since the dimension d is relatively small in our test problem). For comparison, we also tested gradient descent (GD) and the classical BFGS quasi-Newton method, where we use line search to obtain their best performance (Nocedal and Wright, 2006).



(a) Convergence by iteration (b) Convergence by gradient evaluations (c) Histogram of gradient evaluations

Figure 1: Numerical results for a regularized logistic regression problem.

From Figure 1(a), we observe that GD converges to the optimal solution at a slow linear rate, while QNPE and BFGS can achieve a high accuracy in much fewer iterations. We also illustrated our theoretical bound of $(1 + c\sqrt{k})^{-k}$ with a manually tuned parameter c, which matches well with the empirical performance of QNPE. Due to the use of line search, in Figure 1(b) we also compare these algorithms in terms of the number of gradient evaluations. Note that the line search scheme in GD only queries the function value at the new point, and thus it requires exactly one gradient evaluation per iteration. As a result, while QNPE still converges faster than GD, the relative performance gap becomes smaller. On the other hand, we remark that the number of gradient evaluations per iteration for QNPE is still small as guaranteed by Theorem 14. Indeed, as shown in the histogram in Figure 1(c), most of the iterations evaluate 2 or 3 gradients and the average is no more than 3, which we observe consistently across different settings. Finally, we note that BFGS with line search outperforms all the other considered methods in our experiments, despite the fact that its finite-time complexity bound is still lacking. Hence, establishing a global non-asymptotic convergence rate for BFGS is an interesting open problem to explore.

6. Conclusion

We proposed the quasi-Newton proximal extragradient (QNPE) method for unconstrained minimization problems. We showed that QNPE converges at an explicit non-asymptotic superlinear rate of $(1 + \Omega(\sqrt{k}))^{-k}$. Moreover, if N_ϵ denotes the number of iterations to find an ϵ-accurate solution, we showed that the number of gradient evaluations is bounded by $3N_\epsilon$, while the number of matrix-vector product evaluations is bounded by $\mathcal{O}(N_\epsilon\sqrt{\frac{L_1}{\mu}}\log\frac{L_1 N_\epsilon^2 d}{\mu\epsilon})$. To the best of our knowledge, this is the first quasi-Newton method with an explicit global superlinear convergence rate.

Acknowledgments

This work is supported in part by NSF Grants 2007668, 2019844, and 2112471, ARO Grant W911NF2110226, the Machine Learning Lab (MLL) at UT Austin, and the Wireless Networking and Communications Group (WNCG) Industrial Affiliates Program. The authors would also like to thank Anton Rodomanov and the anonymous reviewers for their comments on the first draft of the paper.

References

Charles G Broyden. The convergence of single-rank quasi-Newton methods. *Mathematics of Computation*, 24(110):365–382, 1970.

Charles George Broyden, John E Dennis Jr, and Jorge J Moré. On the local and superlinear convergence of quasi-Newton methods. *IMA Journal of Applied Mathematics*, 12(3):223–245, 1973.

Richard H Byrd, Jorge Nocedal, and Ya-Xiang Yuan. Global convergence of a class of quasi-Newton methods on convex problems. *SIAM Journal on Numerical Analysis*, 24(5):1171–1190, 1987.

Richard H Byrd, Humaid Fayez Khalfan, and Robert B Schnabel. Analysis of a symmetric rank-one trust region method. *SIAM Journal on Optimization*, 6(4):1025–1039, 1996.

Andrew R Conn, Nicholas IM Gould, and Ph L Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical programming*, 50(1):177–195, 1991.

W. C. Davidon. Variable metric method for minimization. Techinical Report ANL-5990, Argonne National Laboratory, Argonne, IL, 1959.

John E Dennis and Jorge J Moré. A characterization of superlinear convergence and its application to quasi-Newton methods. *Mathematics of computation*, 28(126):549–560, 1974.

Laurence Charles Ward Dixon. Variable metric algorithms: necessary and sufficient conditions for identical behavior of nonquadratic functions. *Journal of Optimization Theory and Applications*, 10(1):34–40, 1972.

Roger Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

Roger Fletcher and Michael JD Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

David Chin-Lung Fong. *Minimum-Residual Methods for Sparse Least-Squares Using Golub-Kahan Bidiagonalization*. PhD thesis, Stanford University, 2011.

Donald Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

Anne Greenbaum. *Iterative methods for solving linear systems*. SIAM, 1997.

Qiujiang Jin and Aryan Mokhtari. Non-asymptotic superlinear convergence of standard quasi-Newton methods. *Mathematical Programming*, 2022.

Qiujiang Jin, Alec Koppel, Ketan Rajawat, and Aryan Mokhtari. Sharpened quasi-Newton methods: Faster superlinear rate and larger local convergence neighborhood. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10228–10250. PMLR, 2022.

H Fayez Khalfan, Richard H Byrd, and Robert B Schnabel. A theoretical and experimental study of the symmetric rank-one update. *SIAM Journal on Optimization*, 3(1):1–24, 1993.

G. Korpelevich. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976. In Russian; English translation in Matekon.

J. Kuczyński and H. Woźniakowski. Estimating the Largest Eigenvalue by the Power and Lanczos Algorithms with a Random Start. *SIAM Journal on Matrix Analysis and Applications*, 13(4): 1094–1122, 1992.

Dachao Lin, Haishan Ye, and Zhihua Zhang. Greedy and random quasi-newton methods with faster explicit superlinear convergence. *Advances in Neural Information Processing Systems*, 34:6646–6657, 2021.

B. Martinet. Brève communication. régularisation d'inéquations variationnelles par approximations successives. *ESIAM Mathematical Modelling and Numerical Analysis*, 4(R3):154–158, 1970.

Zakaria Mhammedi. Efficient projection-free online convex optimization with membership oracle. In *Proceedings of Thirty Fifth Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 5314–5390. PMLR, 2022.

Renato D. C. Monteiro and B. F. Svaiter. On the complexity of the hybrid proximal extragradient method for the iterates and the ergodic mean. *SIAM Journal on Optimization*, 20(6):2755–2787, 2010.

Renato DC Monteiro and Benar F Svaiter. Iteration-complexity of a Newton proximal extragradient method for monotone variational inequalities and inclusion problems. *SIAM Journal on Optimization*, 22(3):914–935, 2012.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Science+Business Media, LLC, 2006.

Dianne P. O'Leary. A Matlab implementation of a MINPACK line search algorithm by Jorge J. Moré and David J. Thuente. http://www.cs.umd.edu/users/oleary/software/, 1991.

M. J. D. Powell. On the convergence of the variable metric algorithm. *IMA Journal of Applied Mathematics*, 7(1):21–36, 1971.

M. J. D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In *Nonlinear Programming*, volume IX of *SIAM-AMS Proceedings*, Philadelphia, 1976. Society for Industrial and Applied Mathematics.

R. Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898, 1976.

Anton Rodomanov and Yurii Nesterov. Rates of superlinear convergence for classical quasi-newton methods. *Mathematical Programming*, 2021a.

Anton Rodomanov and Yurii Nesterov. Greedy quasi-newton methods with explicit superlinear convergence. *SIAM Journal on Optimization*, 31(1):785–811, 2021b.

Anton Rodomanov and Yurii Nesterov. New results on superlinear convergence of classical quasi-newton methods. *Journal of Optimization Theory and Applications*, 188(3):744–769, 2021c.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

David F Shanno. Conditioning of quasi-Newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

Mikhail V Solodov and Benar F Svaiter. A hybrid approximate extragradient–proximal point algorithm using the enlargement of a maximal monotone operator. *Set-Valued Analysis*, 7(4):323–345, 1999.

Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low noise and fast rates. *Advances in neural information processing systems*, 23, 2010.

Haishan Ye, Dachao Lin, Xiangyu Chang, and Zhihua Zhang. Towards explicit superlinear convergence rate for sr1. *Mathematical Programming*, pages 1–31, 2022.

Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

Appendix A. Missing Proofs in Section 3

A.1. Proof of Proposition 1

In this section, we provide the proof of Proposition 1. We also prove an additional result in (19), which will be useful later in the proof of Lemma 13.

Proposition 1 *Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the iterates generated by (8), (9), and (10) where $\alpha_1 + \alpha_2 < 1$. If f is μ-strongly convex, then we have*

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_k - \mathbf{x}^*\|^2(1 + 2\eta_k\mu)^{-1}. \tag{18}$$

Moreover, we have

$$\sum_{k=0}^{N-1} \|\hat{\mathbf{x}}_k - \mathbf{x}_k\|^2 \leq \frac{1}{1-\alpha_1-\alpha_2}\|\mathbf{x}_0 - \mathbf{x}^*\|^2. \tag{19}$$

Proof To simplify the notation, let $\alpha = \alpha_1 + \alpha_2 \in (0,1)$. For any $\mathbf{x} \in \mathbb{R}^d$, we can write

$$\eta_k\langle \nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}\rangle = \langle \hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k \nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}\rangle + \langle \mathbf{x}_k - \hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k - \mathbf{x}\rangle. \tag{20}$$

To begin with, by using the triangle inequality and the conditions in (8) and (9), we observe that

$$\begin{aligned}
&\|\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k\nabla f(\hat{\mathbf{x}}_k)\| \\
&= \|\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k(\nabla f(\mathbf{x}_k) + \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k)) + \eta_k\nabla f(\hat{\mathbf{x}}_k) - \eta_k(\nabla f(\mathbf{x}_k) + \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k))\| \\
&\leq \|\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k(\nabla f(\mathbf{x}_k) + \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k))\| + \eta_k\|\nabla f(\hat{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k)\| \\
&\leq (\alpha_1 + \alpha_2)\|\hat{\mathbf{x}}_k - \mathbf{x}_k\| = \alpha\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|.
\end{aligned} \tag{21}$$

Thus, we can bound the first term in (20) by

$$\begin{aligned}
\langle \hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k\nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}\rangle &\leq \|\hat{\mathbf{x}}_k - \mathbf{x}_k + \eta_k\nabla f(\hat{\mathbf{x}}_k)\|\|\hat{\mathbf{x}}_k - \mathbf{x}\| \\
&\leq \alpha\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|\|\hat{\mathbf{x}}_k - \mathbf{x}\| \\
&\leq \frac{\alpha}{2}\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|^2 + \frac{\alpha}{2}\|\hat{\mathbf{x}}_k - \mathbf{x}\|^2,
\end{aligned} \tag{22}$$

where the first inequality is due to Cauchy-Schwarz inequality, the second inequality is due to (21), and the last inequality is due to Young's inequality. Moreover, for the second term in (20), we use the three-point equality to get

$$\langle \mathbf{x}_k - \hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k - \mathbf{x}\rangle = \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}\|^2 - \frac{1}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 - \frac{1}{2}\|\hat{\mathbf{x}}_k - \mathbf{x}\|^2. \tag{23}$$

By combining (20), (22) and (23), we obtain that

$$\eta_k\langle \nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}\rangle \leq \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}\|^2 - \frac{1-\alpha}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 - \frac{1-\alpha}{2}\|\hat{\mathbf{x}}_k - \mathbf{x}\|^2. \tag{24}$$

Furthermore, by the update rule in (10), we can write $\eta_k\nabla f(\hat{\mathbf{x}}_k) = \mathbf{x}_k - \mathbf{x}_{k+1} + 2\eta_k\mu(\hat{\mathbf{x}}_k - \mathbf{x}_{k+1})$. This implies that, for any $\mathbf{x} \in \mathbb{R}^d$,

$$\begin{aligned}
&\eta_k\langle \nabla f(\hat{\mathbf{x}}_k), \mathbf{x}_{k+1} - \mathbf{x}\rangle \\
&= \langle \mathbf{x}_k - \mathbf{x}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}\rangle + 2\eta_k\mu\langle \hat{\mathbf{x}}_k - \mathbf{x}_{k+1}, \mathbf{x}_{k+1} - \mathbf{x}\rangle \\
&= \frac{\|\mathbf{x}_k - \mathbf{x}\|^2}{2} - \frac{\|\mathbf{x}_k - \mathbf{x}_{k+1}\|^2}{2} - \frac{1+2\eta_k\mu}{2}\|\mathbf{x}_{k+1} - \mathbf{x}\|^2 + \eta_k\mu\|\hat{\mathbf{x}}_k - \mathbf{x}\|^2 - \eta_k\mu\|\hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\|^2,
\end{aligned} \tag{25}$$

where the last equality comes from the three-point equality. Thus, by combining (24) with $\mathbf{x} = \mathbf{x}_{k+1}$ and (25) with $\mathbf{x} = \mathbf{x}^*$, we get

$$\begin{aligned}\eta_k\langle \nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}^*\rangle &= \eta_k\langle \nabla f(\hat{\mathbf{x}}_k), \mathbf{x}_{k+1} - \mathbf{x}^*\rangle + \eta_k\langle \nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\rangle \\ &\le \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}^*\|^2 - \cancel{\frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_{k+1}\|^2} - \frac{1+2\eta_k\mu}{2}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 \\ &\quad + \eta_k\mu\|\hat{\mathbf{x}}_k - \mathbf{x}^*\|^2 - \eta_k\mu\|\hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\|^2 \\ &\quad + \cancel{\frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_{k+1}\|^2} - \frac{1-\alpha}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 - \frac{1-\alpha}{2}\|\hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\|^2.\end{aligned} \tag{26}$$

Since f is μ-strongly convex, we have

$$\langle \nabla f(\hat{\mathbf{x}}_k), \hat{\mathbf{x}}_k - \mathbf{x}^*\rangle = \langle \nabla f(\hat{\mathbf{x}}_k) - \nabla f(\mathbf{x}^*), \hat{\mathbf{x}}_k - \mathbf{x}^*\rangle \ge \mu\|\hat{\mathbf{x}}_k - \mathbf{x}^*\|^2. \tag{27}$$

Combining (26) and (27) and rearranging the terms, we obtain

$$\begin{aligned}\frac{1+2\eta_k\mu}{2}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 &\le \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}^*\|^2 - \frac{1-\alpha}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 - \left(\frac{1-\alpha}{2} + \eta_k\mu\right)\|\hat{\mathbf{x}}_k - \mathbf{x}_{k+1}\|^2 \\ &\le \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}^*\|^2 - \frac{1-\alpha}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2.\end{aligned} \tag{28}$$

Since $\alpha < 1$, the last term in (28) is negative and (18) follows immediately. Moreover, since $\frac{1}{2}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 \le \frac{1+2\eta_k\mu}{2}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2$, we obtain from (28) that

$$\frac{1-\alpha}{2}\|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2 \le \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}^*\|^2 - \frac{1}{2}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2. \tag{29}$$

by summing (29) over $k = 0, 1, \dots, N-1$, we can get

$$\sum_{k=0}^{N-1}\frac{1-\alpha}{2}\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|^2 \le \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|^2 - \frac{1}{2}\|\mathbf{x}_N - \mathbf{x}^*\|^2 \le \frac{1}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|^2,$$

which implies (19). The proof is complete. ■

A.2. Proof of Lemma 3

If $k \notin \mathcal{B}$, by definition, the line search scheme accepts the initial trial step size at the k-th iteration, which means $\eta_k = \sigma_k$. Otherwise, if $k \in \mathcal{B}$, recall that $\tilde{\mathbf{x}}_k$ is the last rejected point in the line search scheme, which is computed from (13) using step size $\tilde{\eta}_k = \eta_k/\beta$. This means that the pair $(\tilde{\mathbf{x}}_k, \tilde{\eta}_k)$ does not satisfy (12), i.e., $\tilde{\eta}_k\|\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\| > \alpha_2\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|$, which implies

$$\eta_k = \beta\tilde{\eta}_k > \frac{\alpha_2\beta\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|}{\|\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\|}.$$

This proves the first inequality in (14). To prove the second inequality, recall from (11) that $\hat{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}_k$ can be regarded as the inexact solution of the linear system of equations

$$(\mathbf{I} + \eta_k\mathbf{B}_k)(\mathbf{x} - \mathbf{x}_k) = -\eta_k\nabla f(\mathbf{x}_k) \quad \text{and} \quad (\mathbf{I} + \tilde{\eta}_k\mathbf{B}_k)(\mathbf{x} - \mathbf{x}_k) = -\tilde{\eta}_k\nabla f(\mathbf{x}_k),$$

respectively. Define $\hat{\mathbf{x}}_k^* = \mathbf{x}_k - \eta_k(\mathbf{I} + \eta_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k)$ and $\tilde{\mathbf{x}}_k^* = \mathbf{x}_k - \tilde{\eta}_k(\mathbf{I} + \tilde{\eta}_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k)$, i.e., the exact solutions of the above linear systems. Since $(\hat{\mathbf{x}}_k, \eta_k)$ and $(\tilde{\mathbf{x}}_k, \tilde{\eta}_k)$ satisfy the condition in (11), we have

$$\|(\mathbf{I} + \eta_k\mathbf{B}_k)(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^*)\| \leq \alpha_1\|\hat{\mathbf{x}}_k - \mathbf{x}_k\| \quad \text{and} \quad \|(\mathbf{I} + \tilde{\eta}_k\mathbf{B}_k)(\tilde{\mathbf{x}}_k - \tilde{\mathbf{x}}_k^*)\| \leq \alpha_1\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|. \tag{30}$$

We divide the proof of the second inequality in (14) into the following three steps. First, we show that

$$(1-\alpha_1)\|\hat{\mathbf{x}}_k - \mathbf{x}_k\| \leq \|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\| \leq (1+\alpha_1)\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|, \tag{31}$$

$$(1-\alpha_1)\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\| \leq \|\tilde{\mathbf{x}}_k^* - \mathbf{x}_k\| \leq (1+\alpha_1)\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|. \tag{32}$$

In the following, we will only prove (31), since the proof of (32) follows similarly. Using the fact that $\mathbf{B}_k \in \mathbb{S}_+^d$, we have $\|(\mathbf{I} + \eta_k\mathbf{B}_k)(\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^*)\| \geq \|\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^*\|$. Hence, combining this with (30), we get $\|\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^*\| \leq \alpha_1\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|$. It then follows from the triangle inequality that

$$\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\| \leq \|\hat{\mathbf{x}}_k - \mathbf{x}_k\| + \|\hat{\mathbf{x}}_k^* - \hat{\mathbf{x}}_k\| \leq (1+\alpha_1)\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|,$$
$$\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\| \geq \|\hat{\mathbf{x}}_k - \mathbf{x}_k\| - \|\hat{\mathbf{x}}_k^* - \hat{\mathbf{x}}_k\| \geq (1-\alpha_1)\|\hat{\mathbf{x}}_k - \mathbf{x}_k\|,$$

which proves (31). Next, we show that

$$\|\tilde{\mathbf{x}}_k^* - \mathbf{x}_k\| \leq \frac{1}{\beta}\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\|. \tag{33}$$

To see this, we can compute

$$\|\tilde{\mathbf{x}}_k^* - \mathbf{x}_k\| = \|\tilde{\eta}_k(\mathbf{I} + \tilde{\eta}_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k)\| \leq \|\tilde{\eta}_k(\mathbf{I} + \eta_k\mathbf{B}_k)^{-1}\nabla f(\mathbf{x}_k)\| = \frac{\tilde{\eta}_k}{\eta_k}\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\| = \frac{1}{\beta}\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\|,$$

where we used the fact that $\mathbf{I} + \tilde{\eta}_k\mathbf{B}_k \succeq \mathbf{I} + \eta_k\mathbf{B}_k$ in the first inequality. Finally, by combining (31), (32), and (33), we obtain

$$\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\| \leq \frac{1}{1-\alpha_1}\|\tilde{\mathbf{x}}_k^* - \mathbf{x}_k\| \leq \frac{1}{\beta(1-\alpha_1)}\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\| \leq \frac{1+\alpha_1}{\beta(1-\alpha_1)}\|\hat{\mathbf{x}}_k^* - \mathbf{x}_k\|.$$

This completes the proof.

A.3. Proof of Lemma 4

Recall that in Lemma 3, we proved that $\eta_k = \sigma_k$ if $k \notin \mathcal{B}$ and $\eta_k > \frac{\alpha_2\beta\|\mathbf{s}_k\|}{\|\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\|}$ otherwise, where $\mathbf{y}_k \triangleq \nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k)$ and $\mathbf{s}_k \triangleq \tilde{\mathbf{x}}_k - \mathbf{x}_k$. Using the observations above, we can write

$$\begin{aligned}\sum_{k=0}^{N-1}\frac{1}{\eta_k^2} = \sum_{k\notin\mathcal{B}}\frac{1}{\eta_k^2} + \sum_{k\in\mathcal{B}}\frac{1}{\eta_k^2} &\leq \sum_{k\notin\mathcal{B}}\frac{1}{\sigma_k^2} + \frac{1}{\alpha_2^2\beta^2}\sum_{k\in\mathcal{B}}\frac{\|\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\|^2}{\|\mathbf{s}_k\|^2} \\ &= \frac{1}{\sigma_0^2} + \beta^2\sum_{k\notin\mathcal{B},k\geq 1}\frac{1}{\eta_{k-1}^2} + \frac{1}{\alpha_2^2\beta^2}\sum_{k\in\mathcal{B}}\frac{\|\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k\|^2}{\|\mathbf{s}_k\|^2},\end{aligned} \tag{34}$$

where we used $\sigma_k = \eta_{k-1}/\beta$ for $k \geq 1$ in the last equality. Since we have

$$\sum_{k\notin\mathcal{B},k\geq 1}\frac{1}{\eta_{k-1}^2} \leq \sum_{k=1}^{N-1}\frac{1}{\eta_{k-1}^2} \leq \sum_{k=0}^{N-1}\frac{1}{\eta_k^2},$$

by rearranging and simplifying the terms in (34), we arrive at the inequality in (16).

Appendix B. Proof of Theorem 10

In this section, we formally prove Lemmas 11-13 used in Theorem 10. As discussed in the main text, throughout the proof, we assume that every call of ExtEvec is successful, which happens with probability at least $1-p$. Specifically, since the ExtEvec oracle has a failure probability of $q_t = p/2.5(t+1)\log^2(t+1)$ in the t-th round, we can use the union bound to upper bound the total failure probability by

$$\sum_{t=1}^{T-1} q_t = \frac{p}{2.5}\sum_{t=2}^{T}\frac{1}{t\log^2 t} \leq \frac{p}{2.5}\sum_{t=2}^{\infty}\frac{1}{t\log^2 t} \leq \frac{p}{2.5}\left(\frac{1}{2\log^2 2} + \int_2^{+\infty}\frac{1}{t\log^2 t}\,dt\right) \leq p.$$

As a result, we always have $\mathbf{B}_t \in \mathcal{Z}'$, i.e., the eigenvalue of $\mathbf{B}_t$ is bounded between $\frac{\mu}{2}$ and $L_1 + \frac{\mu}{2}$. This property will be used in the proof of Lemma 11 and Lemma 12.

B.1. Proof of Lemma 11

We present the general version of Lemma 11 below that applies for any $\alpha_2 \in (0,1)$ and $\beta \in (0,1)$.

Lemma 11 *For any $k \geq 0$, we have $\eta_k \geq \alpha_2\beta/L_1$.*

Proof We first establish that $\eta_k \geq \alpha_2\beta/L_1$ for $k \in \mathcal{B}$. To see this, suppose $k \in \mathcal{B}$ and recall from Lemma 3 that

$$\eta_k > \frac{\alpha_2\beta\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|}{\|\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\|}. \tag{35}$$

By the fundamental theorem of calculus, we can write $\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) = \bar{\mathbf{H}}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)$, where $\bar{\mathbf{H}}_k = \int_0^1 \nabla^2 f(t\tilde{\mathbf{x}}_k + (1-t)\mathbf{x}_k)\,dt$. Since we have $\mu\mathbf{I} \preceq \nabla^2 f(\mathbf{x}) \preceq L_1\mathbf{I}$ for all $\mathbf{x} \in \mathbb{R}^d$ by Assumption 1, we get $\mu\mathbf{I} \preceq \bar{\mathbf{H}}_k \preceq L_1\mathbf{I}$. Moreover, since $\frac{\mu}{2}\mathbf{I} \preceq \mathbf{B}_k \preceq (L_1 + \frac{\mu}{2})\mathbf{I}$, we further have $(-L_1 + \frac{\mu}{2})\mathbf{I} \preceq \bar{\mathbf{H}}_k - \mathbf{B}_k \preceq (L_1 - \frac{\mu}{2})\mathbf{I}$, which implies $\|\bar{\mathbf{H}}_k - \mathbf{B}_k\|_{\mathrm{op}} \leq L_1 - \frac{\mu}{2} \leq L_1$. Thus, we have

$$\|\nabla f(\tilde{\mathbf{x}}_k) - \nabla f(\mathbf{x}_k) - \mathbf{B}_k(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\| = \|(\bar{\mathbf{H}}_k - \mathbf{B}_k)(\tilde{\mathbf{x}}_k - \mathbf{x}_k)\| \leq L_1\|\tilde{\mathbf{x}}_k - \mathbf{x}_k\|,$$

which proves that $\eta_k > \alpha_2\beta/L_1$ from (35).

Now we can prove that $\eta_k \geq \alpha_2\beta/L_1$ for all $k \geq 0$ by induction. To show that this holds true for $k = 0$, we distinguish two cases. If $0 \notin \mathcal{B}$, then we have $\eta_0 = \sigma_0 > \alpha_2\beta/L_1$ by our choice of σ_0. Otherwise, if $0 \in \mathcal{B}$, then it directly follows from our result in the previous paragraph. Moreover, assume that $\eta_{l-1} \geq \alpha_2\beta/L_1$ where $l \geq 1$. Similarly, we again distinguish two cases: if $l \notin \mathcal{B}$, then we have $\eta_l = \sigma_l = \eta_{l-1}/\beta > \alpha_2/L_1 > \alpha_2\beta/L_1$; otherwise, if $l \in \mathcal{B}$, it follows from the result above that $\eta_l \geq \alpha_2\beta/L_1$. This completes the induction. ■

B.2. Proof of Lemma 12

Recall that our Hessian approximation update in Subroutine 2 is a direct instantiation of the general projection-free online learning algorithm described in Section 3.2.1. Therefore, we first present the regret analysis of the general algorithm in Lemma 15. For completeness, the pseudocode of the general algorithm is also given in Algorithm 2.

Algorithm 2 Projection-Free Online Learning

```
Input: Initial point $\mathbf{w}_0 \in \mathcal{B}_R(0)$, step size $\rho > 0$, $\delta > 0$
for $t = 0, 1, \ldots T - 1$ do
    Query the oracle $(\gamma_t, \mathbf{s}_t) \leftarrow \mathsf{SEP}(\mathbf{w}_t; \delta)$
    if $\gamma_t \leq 1$ then    # Case I: we have $\mathbf{w}_t \in (1+\delta)\mathcal{C}$
        Set $\mathbf{x}_t \leftarrow \mathbf{w}_t$ and play the action $\mathbf{x}_t$
        Receive the loss $\ell_t(\mathbf{x}_t)$ and the gradient $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$
        Set $\tilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t$
    else    # Case II: we have $\mathbf{w}_t/\gamma_t \in (1+\delta)\mathcal{C}$
        Set $\mathbf{x}_t \leftarrow \mathbf{w}_t/\gamma_t$ and play the action $\mathbf{x}_t$
        Receive the loss $\ell_t(\mathbf{x}_t)$ and the gradient $\mathbf{g}_t = \nabla \ell_t(\mathbf{x}_t)$
        Set $\tilde{\mathbf{g}}_t \leftarrow \mathbf{g}_t + \max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t \rangle\}\mathbf{s}_t$
    end if
    Update $\mathbf{w}_{t+1} \leftarrow \frac{R}{\max\{\|\mathbf{w}_t - \rho\tilde{\mathbf{g}}_t\|_2, R\}}(\mathbf{w}_t - \rho\tilde{\mathbf{g}}_t)$    # Euclidean projection onto $\mathcal{B}_R(0)$
end for
```

Lemma 15 *Let $\{\mathbf{x}_t\}_{t=0}^{T-1}$ be the iterates generated by Algorithm 2. Then we have $\mathbf{x}_t \in (1+\delta)\mathcal{C}$ for $t = 0, 1, \ldots, T-1$. Also, for any $\mathbf{x} \in \mathcal{C}$, we have*

$$\langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{x} \rangle \leq \langle \tilde{\mathbf{g}}_t, \mathbf{w}_t - \mathbf{x} \rangle \leq \frac{1}{2\rho}\|\mathbf{w}_t - \mathbf{x}\|_2^2 - \frac{1}{2\rho}\|\mathbf{w}_{t+1} - \mathbf{x}\|_2^2 + \frac{\rho}{2}\|\tilde{\mathbf{g}}_t\|_2^2, \tag{36}$$

and

$$\|\tilde{\mathbf{g}}_t\| \leq \|\mathbf{g}_t\| + |\langle \mathbf{g}_t, \mathbf{x}_t \rangle| \|\mathbf{s}_t\|. \tag{37}$$

Proof We distinguish two cases depending on the outcome of $\mathsf{SEP}(\mathbf{w}_t; \delta)$.

- If $\gamma_t \leq 1$, By Definition 7 we have $\mathbf{w}_t \in (1+\delta)\mathcal{C}$. According to Algorithm 2, we have $\mathbf{x}_t = \mathbf{w}_t \in (1+\delta)\mathcal{C}$ and $\tilde{\mathbf{g}}_t = \mathbf{g}_t$, and thus the first inequality in (36) and the inequality in (37) trivially hold.

- Otherwise, if $\gamma_t > 1$, By Definition 7 we have $\mathbf{w}_t/\gamma_t \in (1+\delta)\mathcal{C}$ and $\langle \mathbf{s}_t, \mathbf{w}_t - \mathbf{x} \rangle \geq \gamma_t - 1$ $\forall \mathbf{x} \in \mathcal{C}$. According to Algorithm 2, we have $\mathbf{x}_t = \mathbf{w}_t/\gamma_t \in (1+\delta)\mathcal{C}$ and $\tilde{\mathbf{g}}_t = \mathbf{g}_t + \max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t \rangle\}\mathbf{s}_t$. To prove the first inequality in (36), note that for any $\mathbf{x} \in \mathcal{C}$,

$$\begin{aligned}
\langle \tilde{\mathbf{g}}_t, \mathbf{w}_t - \mathbf{x} \rangle &= \langle \mathbf{g}_t + \max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t \rangle\}\mathbf{s}_t, \mathbf{w}_t - \mathbf{x} \rangle \\
&= \langle \mathbf{g}_t, \gamma_t \mathbf{x}_t - \mathbf{x} \rangle + \max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t \rangle\}\langle \mathbf{s}_t, \mathbf{w}_t - \mathbf{x} \rangle \\
&\geq \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{x} \rangle + (\gamma_t - 1)\langle \mathbf{g}_t, \mathbf{x}_t \rangle + (\gamma_t - 1)\max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t \rangle\} \\
&\geq \langle \mathbf{g}_t, \mathbf{x}_t - \mathbf{x} \rangle,
\end{aligned}$$

  where we used $\langle \mathbf{s}_t, \mathbf{w}_t - \mathbf{x} \rangle \geq \gamma_t - 1$ in the first inequality. Also, by the triangle inequality we obtain

$$\|\tilde{\mathbf{g}}_t\|_2 = \|\mathbf{g}_t + \max\{0, -\langle \mathbf{g}_t, \mathbf{x}_t \rangle\}\mathbf{s}_t\|_2 \leq \|\mathbf{g}_t\|_2 + |\langle \mathbf{g}_t, \mathbf{x}_t \rangle| \|\mathbf{s}_t\|_2,$$

  which proves (37).

Finally, from the update rule of $\mathbf{w}_{t+1}$, for any $\mathbf{x} \in \mathcal{C} \subset \mathcal{B}_R(0)$ we have $\langle \mathbf{w}_t - \rho\tilde{\mathbf{g}}_t - \mathbf{w}_{t+1}, \mathbf{w}_{t+1} - \mathbf{x}\rangle \geq 0$, which further implies that

$$\begin{aligned}\langle \tilde{\mathbf{g}}_t, \mathbf{w}_t - \mathbf{x}\rangle &\leq \langle \tilde{\mathbf{g}}_t, \mathbf{w}_t - \mathbf{w}_{t+1}\rangle + \frac{1}{\rho}\langle \mathbf{w}_t - \mathbf{w}_{t+1}, \mathbf{w}_{t+1} - \mathbf{x}\rangle \\ &= \langle \tilde{\mathbf{g}}_t, \mathbf{w}_t - \mathbf{w}_{t+1}\rangle + \frac{1}{2\rho}\|\mathbf{w}_t - \mathbf{x}\|_2^2 - \frac{1}{2\rho}\|\mathbf{w}_{t+1} - \mathbf{x}\|_2^2 - \frac{1}{2\rho}\|\mathbf{w}_t - \mathbf{w}_{t+1}\|_2^2 \\ &\leq \frac{1}{2\rho}\|\mathbf{w}_t - \mathbf{x}\|_2^2 - \frac{1}{2\rho}\|\mathbf{w}_{t+1} - \mathbf{x}\|_2^2 + \frac{\rho}{2}\|\tilde{\mathbf{g}}_t\|_2^2. \end{aligned} \tag{38}$$

This proves the second inequality in (36). ■

Next, we present the following lemma showing a smooth property of the loss function ℓ_k. It is similar to the standard inequality $\frac{1}{2L_1}\|\nabla g(\mathbf{x})\|^2 \leq g(\mathbf{x}) - g^*$ for a L_1-smooth function g, where g^* denotes the minimum of g. This will be the key to proving a constant upper bound on the cumulative loss incurred by Subroutine 2.

Lemma 16 *Recall the loss function ℓ_k defined in (17). For $k \in \mathcal{B}$, we have*

$$\nabla \ell_k(\mathbf{B}) = \frac{1}{2\|\mathbf{s}_k\|^2}\left(-\mathbf{s}_k(\mathbf{y}_k - \mathbf{B}\mathbf{s}_k)^{\mathsf{T}} - (\mathbf{y}_k - \mathbf{B}\mathbf{s}_k)\mathbf{s}_k^{\mathsf{T}}\right). \tag{39}$$

Moreover, for any $\mathbf{B} \in \mathbb{S}^d$, it holds that

$$\|\nabla \ell_k(\mathbf{B})\|_F \leq \|\nabla \ell_k(\mathbf{B})\|_* \leq \sqrt{2\ell_k(\mathbf{B})}, \tag{40}$$

where $\|\cdot\|_F$ and $\|\cdot\|_*$ denote the Frobenius norm and the nuclear norm, respectively.

Proof The expression in (39) follows from direct calculation. The first inequality in (40) follows from the fact that $\|\mathbf{A}\|_F \leq \|\mathbf{A}\|_*$ for any matrix $\mathbf{A} \in \mathbb{S}^d$. For the second inequality, note that

$$\begin{aligned}\|\nabla \ell_t(\mathbf{B})\|_* &\leq \frac{1}{2\|\mathbf{s}_t\|^2}\left(\|\mathbf{s}_t(\mathbf{y}_t - \mathbf{B}\mathbf{s}_t)^{\mathsf{T}}\|_* + \|(\mathbf{y}_t - \mathbf{B}\mathbf{s}_t)\mathbf{s}_t^{\mathsf{T}}\|_*\right) \\ &\leq \frac{1}{\|\mathbf{s}_t\|^2}\|\mathbf{y}_t - \mathbf{B}\mathbf{s}_t\|\|\mathbf{s}_t\| = \frac{\|\mathbf{y}_t - \mathbf{B}\mathbf{s}_t\|}{\|\mathbf{s}_t\|} = \sqrt{2\ell_t(\mathbf{B})},\end{aligned}$$

where in the first inequality we used the triangle inequality, and in the second inequality we used the fact that the rank-one matrix $\mathbf{u}\mathbf{v}^\top$ has only one nonzero singular value $\|\mathbf{u}\|\|\mathbf{v}\|$. ■

Now we are ready to present the proof of Lemma 12. By letting $\mathbf{x}_t = \hat{\mathbf{B}}_t$, $\mathbf{x} = \hat{\mathbf{B}} \triangleq \frac{2}{L_1-\mu}(\mathbf{H} - \frac{L_1+\mu}{2}\mathbf{I})$, $\mathbf{g}_t = \mathbf{G}_t \triangleq \frac{2}{L_1-\mu}\nabla\ell_t(\mathbf{B}_t)$, $\tilde{\mathbf{g}}_t = \tilde{\mathbf{G}}_t$, $\mathbf{w}_t = \mathbf{W}_t$ in Lemma 15, we obtain:

(i) $\hat{\mathbf{B}}_t \in (1+\delta)\mathcal{C}$, which means $\|\hat{\mathbf{B}}_t\|_{\text{op}} \leq 1 + \delta \leq 2$ since $\delta \leq 1$.

(ii) It holds that

$$\langle \mathbf{G}_t, \hat{\mathbf{B}}_t - \hat{\mathbf{B}}\rangle \leq \frac{1}{2\rho}\|\mathbf{W}_t - \hat{\mathbf{B}}\|_F^2 - \frac{1}{2\rho}\|\mathbf{W}_{t+1} - \hat{\mathbf{B}}\|_F^2 + \frac{\rho}{2}\|\tilde{\mathbf{G}}_t\|_F^2, \tag{41}$$

$$\|\tilde{\mathbf{G}}_t\|_F \leq \|\mathbf{G}_t\|_F + |\langle \mathbf{G}_t, \hat{\mathbf{B}}_t\rangle|\|\mathbf{S}_t\|_F. \tag{42}$$

First, note that $\|\mathbf{S}_t\|_F = 1$ by Definition 9 and $|\langle \mathbf{G}_t, \hat{\mathbf{B}}_t\rangle| \leq \|\mathbf{G}_t\|_* \|\hat{\mathbf{B}}_t\|_{\mathrm{op}} \leq 2\|\mathbf{G}_t\|_*$. Together with (42), we get

$$\|\tilde{\mathbf{G}}_t\|_F \leq \|\mathbf{G}_t\|_F + 2\|\mathbf{G}_t\|_* \leq 3\|\mathbf{G}_t\|_* \leq \frac{6}{L_1 - \mu}\sqrt{2\ell_t(\mathbf{B}_t)}, \tag{43}$$

where we used $\mathbf{G}_t = \frac{2}{L_1+\mu}\nabla \ell_t(\mathbf{B}_t)$ and Lemma 16 in the last inequality. Furthermore, since ℓ_t is convex, we have

$$\ell_t(\mathbf{B}_t) - \ell_t(\mathbf{H}) \leq \langle \nabla \ell_t(\mathbf{B}_t), \mathbf{B}_t - \mathbf{H}\rangle = \left(\frac{L_1 - \mu}{2}\right)^2 \langle \mathbf{G}_t, \hat{\mathbf{B}}_t - \hat{\mathbf{B}}\rangle,$$

where we used $\mathbf{G}_t = \frac{2}{L_1-\mu}\nabla \ell_t(\mathbf{B}_t)$, $\hat{\mathbf{B}}_t \triangleq \frac{2}{L_1-\mu}(\mathbf{B}_t - \frac{L_1+\mu}{2}\mathbf{I})$, and $\hat{\mathbf{B}} \triangleq \frac{2}{L_1-\mu}(\mathbf{H} - \frac{L_1+\mu}{2}\mathbf{I})$. Therefore, by (41) and (43) we get

$$\begin{aligned}\ell_t(\mathbf{B}_t) - \ell_t(\mathbf{H}) &\leq \frac{(L_1-\mu)^2}{8\rho}\|\mathbf{W}_t - \hat{\mathbf{B}}\|_F^2 - \frac{(L_1-\mu)^2}{8\rho}\|\mathbf{W}_{t+1} - \hat{\mathbf{B}}\|_F^2 + \frac{\rho}{2}\left(\frac{L_1-\mu}{2}\right)^2 \|\tilde{\mathbf{G}}_t\|_F^2 \\ &\leq \frac{(L_1-\mu)^2}{8\rho}\|\mathbf{W}_t - \hat{\mathbf{B}}\|_F^2 - \frac{(L_1-\mu)^2}{8\rho}\|\mathbf{W}_{t+1} - \hat{\mathbf{B}}\|_F^2 + 9\rho\ell_t(\mathbf{B}_t).\end{aligned}$$

Since $\rho = 1/18$, by rearranging and simplifying terms in the above inequality, we obtain

$$\ell_t(\mathbf{B}_t) \leq 2\ell_t(\mathbf{H}) + \frac{9(L_1-\mu)^2}{2}\|\mathbf{W}_t - \hat{\mathbf{B}}\|_F^2 - \frac{9(L_1-\mu)^2}{2}\|\mathbf{W}_{t+1} - \hat{\mathbf{B}}\|_F^2.$$

By summing the above inequality from $t = 0$ to $T-1$, we further have

$$\sum_{t=0}^{T-1} \ell_t(\mathbf{B}_t) \leq \frac{9(L_1-\mu)^2}{2}\|\mathbf{W}_0 - \hat{\mathbf{B}}\|_F^2 + 2\sum_{t=0}^{T-1}\ell_t(\mathbf{H}) = 18\|\mathbf{B}_0 - \mathbf{H}\|_F^2 + 2\sum_{t=0}^{T-1}\ell_t(\mathbf{H}),$$

where the last equality is due to $\mathbf{W}_0 \triangleq \frac{2}{L_1-\mu}(\mathbf{B}_0 - \frac{L_1+\mu}{2}\mathbf{I})$ and $\hat{\mathbf{B}} \triangleq \frac{2}{L_1-\mu}(\mathbf{H} - \frac{L_1+\mu}{2}\mathbf{I})$. This completes the proof.

B.3. Proof of Lemma 13

We present the general version of Lemma 13 below that applies for any $\alpha_1, \alpha_2 \in (0,1)$ with $\alpha_1 + \alpha_2 < 1$ and $\beta \in (0,1)$.

Lemma 13 *We have*

$$\sum_{t=0}^{T-1} \ell_t(\mathbf{H}^*) \leq \left(\frac{(1+\alpha_1)^2}{4(1-\alpha_1)^2\beta^2(1-\alpha_1-\alpha_2)} + 1 + \frac{L_1}{2\alpha_2\beta\mu}\right) L_2^2 \|\mathbf{x}_0 - \mathbf{x}^*\|^2.$$

Proof By the fundamental theorem of calculus, we can write $\mathbf{y}_t = \nabla f(\tilde{\mathbf{x}}_t) - \nabla f(\mathbf{x}_t) = \bar{\mathbf{H}}_t(\tilde{\mathbf{x}}_t - \mathbf{x}_t)$, where $\bar{\mathbf{H}}_t = \int_0^1 \nabla^2 f(\mathbf{x}_t + \lambda \mathbf{s}_t)\, d\lambda$. Moreover, we have

$$\begin{aligned}\|\bar{\mathbf{H}}_t - \mathbf{H}^*\|_{\mathrm{op}} \leq \int_0^1 \|(\nabla^2 f(\mathbf{x}_t + \lambda \mathbf{s}_t) - \nabla^2 f(\mathbf{x}^*))\|_{\mathrm{op}}\, d\lambda &\leq L_2 \int_0^1 \|\mathbf{x}_t - \lambda \mathbf{s}_t + \mathbf{x}^*\|\, d\lambda \\ &\leq L_2 \int_0^1 (\|\mathbf{x}_t - \mathbf{x}^*\| + \lambda\|\mathbf{s}_t\|)\, d\lambda \\ &= L_2\Big(\|\mathbf{x}_t - \mathbf{x}^*\| + \frac{1}{2}\|\mathbf{s}_t\|\Big),\end{aligned}$$

where we used Assumption 2 in the second inequality. Therefore, we have $\|\mathbf{y}_t - \mathbf{H}^*\mathbf{s}_t\| = \|(\bar{\mathbf{H}}_t - \mathbf{H}^*)\mathbf{s}_t\| \leq \|\bar{\mathbf{H}}_t - \mathbf{H}^*\|_{\mathrm{op}}\|\mathbf{s}_t\| \leq L_2\|\mathbf{s}_t\|\Big(\|\mathbf{x}_t - \mathbf{x}^*\| + \frac{1}{2}\|\mathbf{s}_t\|\Big)$. This further implies that

$$\sum_{t=0}^{T-1} \ell_t(\mathbf{H}^*) = \sum_{t=0}^{T-1} \frac{\|\mathbf{y}_t - \mathbf{H}^*\mathbf{s}_t\|^2}{2\|\mathbf{s}_t\|^2} \leq \frac{L_2^2}{2}\sum_{t=0}^{T-1}\Big(\|\mathbf{x}_t - \mathbf{x}^*\| + \frac{1}{2}\|\mathbf{s}_t\|\Big)^2$$
$$\leq \frac{L_2^2}{4}\sum_{t=0}^{T-1}\|\mathbf{s}_t\|^2 + L_2^2\sum_{t=0}^{T-1}\|\mathbf{x}_t - \mathbf{x}^*\|^2. \tag{44}$$

To bound the sum $\sum_{t=0}^{T-1}\|\mathbf{s}_t\|^2$, we use Lemma 3 and the inequality in (19) to get

$$\sum_{t=0}^{T-1}\|\mathbf{s}_t\|^2 = \sum_{t=0}^{T-1}\|\tilde{\mathbf{x}}_t - \mathbf{x}_t\|^2 \leq \frac{(1+\alpha_1)^2}{\beta^2(1-\alpha_1)^2}\sum_{t=0}^{T-1}\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|^2 \leq \frac{(1+\alpha_1)^2\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{(1-\alpha_1)^2\beta^2(1-\alpha_1-\alpha_2)}. \tag{45}$$

To bound the sum $\sum_{t=0}^{T-1}\|\mathbf{x}_t - \mathbf{x}^*\|^2$, we use the linear convergence result in Part (a) of Theorem 10:

$$\sum_{t=0}^{T-1}\|\mathbf{x}_t - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_0 - \mathbf{x}^*\|^2\sum_{t=0}^{T-1}\left(1 + \frac{\alpha_2\beta\mu}{L_1}\right)^{-t} \leq \|\mathbf{x}_0 - \mathbf{x}^*\|^2\left(1 + \frac{L_1}{2\alpha_2\beta\mu}\right). \tag{46}$$

Lemma 13 follows immediately from (44), (45), and (46). ■

Appendix C. Characterizing the Computational Cost

In this section, we characterize the computational cost of our QNPE method.

C.1. Implementation of LinearSolver Oracle

In this section, we describe an efficient implementation of the LinearSolver oracle in Definition 2. On a high level, we run the conjugate residual (CR) method (Saad, 2003) to solve the linear system $\mathbf{A}\mathbf{s} = \mathbf{b}$ with $\mathbf{s}_0 = 0$, and returns the iterate $\mathbf{s}_k$ once it satisfies $\|\mathbf{A}\mathbf{s}_k - \mathbf{b}\| \leq \alpha\|\mathbf{s}_k\|$. CR is a Krylov subspace method similar to the better known conjugate gradient (CG) method. In particular, it is designed to minimize the norm of the residual vector $\mathbf{r}_k := \mathbf{b} - \mathbf{A}\mathbf{s}_k$ over the Krylov subspace and thus is more suitable for our purpose. For completeness, the full algorithm is shown in Subroutine 3. Note that in Line 13 we can compute $\mathbf{A}\mathbf{p}_{k+1}$ from $\mathbf{A}\mathbf{r}_{k+1}$ and $\mathbf{A}\mathbf{p}_k$ without an additional matrix-vector product, and hence LinearSolver requires exactly two matrix-vector products in each iteration.

Before presenting the complexity bound of Subroutine 3, we first review some properties of the CR method. In the following, we let $\lambda_{\max}(\mathbf{A})$ and $\lambda_{\min}(\mathbf{A})$ denote the maximum and minimum eigenvalues of $\mathbf{A}$, respectively.

Proposition 17 *Let $\{\mathbf{s}_k\}_{k\geq 0}$ and $\{\mathbf{r}_k\}_{k\geq 0}$ be generated by Subroutine 3. Then the following holds:*

(a) We have

$$\|\mathbf{r}_k\| \leq 2\left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1}\right)^k\|\mathbf{r}_0\|,$$

where $\kappa(\mathbf{A}) = \lambda_{\max}(\mathbf{A})/\lambda_{\min}(\mathbf{A})$ denotes the condition number of $\mathbf{A}$.

Subroutine 3 LinearSolver$(\mathbf{A}, \mathbf{b}; \alpha)$

```
Input: $\mathbf{A} \in \mathbb{S}^d_+$, $\mathbf{b} \in \mathbb{R}^d$, $0 < \alpha < 1$
Initialize: $\mathbf{s}_0 \leftarrow 0$, $\mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{A}\mathbf{s}_0$, $\mathbf{p}_0 \leftarrow \mathbf{r}_0$
for $k = 0, 1, \ldots$ do
    if $\|\mathbf{r}_k\|_2 \le \alpha \|\mathbf{s}_k\|_2$ then
        Return $\mathbf{s}_k$
    end if
    $\alpha_k \leftarrow \langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle / \langle \mathbf{A}\mathbf{p}_k, \mathbf{A}\mathbf{p}_k \rangle$
    $\mathbf{s}_{k+1} \leftarrow \mathbf{s}_k + \alpha_k \mathbf{p}_k$
    $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
    Compute and store $\mathbf{A}\mathbf{r}_{k+1}$
    $\beta_k \leftarrow \langle \mathbf{r}_{k+1}, \mathbf{A}\mathbf{r}_{k+1} \rangle / \langle \mathbf{r}_k, \mathbf{A}\mathbf{r}_k \rangle$
    $\mathbf{p}_{k+1} \leftarrow \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
    Compute and store $\mathbf{A}\mathbf{p}_{k+1} \leftarrow \mathbf{A}\mathbf{r}_{k+1} + \beta_k \mathbf{A}\mathbf{p}_k$
end for
```

Conjugate residual method for solving $\mathbf{A}\mathbf{s} = \mathbf{b}$

(b) We have $\|\mathbf{s}_k\| > \|\mathbf{s}_{k-1}\|$ *for all* $k \ge 1$.

Proof See (Greenbaum, 1997, Section 3.1) for the proof of Part (a) and (Fong, 2011, Theorem 2.1.6) for the proof of Part (b). ■

As a corollary of Proposition 17, we obtain an sufficient condition for $\|\mathbf{r}_k\| \le \alpha \|\mathbf{s}_k\|$.

Lemma 18 *If* $\|\mathbf{r}_k\| \le \alpha \|\mathbf{r}_0\| / \lambda_{\max}(\mathbf{A})$*, then we have* $\|\mathbf{r}_k\| \le \alpha \|\mathbf{s}_k\|$.

Proof From the update rule of Subroutine 3, we can compute that $\mathbf{s}_1 = \frac{\mathbf{b}^\top \mathbf{A}\mathbf{b}}{\|\mathbf{A}\mathbf{b}\|_2^2} \mathbf{b}$, which implies

$$\|\mathbf{s}_1\| = \|\mathbf{b}\| \cdot \frac{\|\mathbf{A}^{1/2}\mathbf{b}\|^2}{(\mathbf{A}^{1/2}\mathbf{b})^\top \mathbf{A} (\mathbf{A}^{1/2}\mathbf{b})} \ge \frac{\|\mathbf{b}\|}{\lambda_{\max}(\mathbf{A})} = \frac{\|\mathbf{r}_0\|}{\lambda_{\max}(\mathbf{A})}.$$

Since $\|\mathbf{s}_k\|$ is strictly increasing (cf. Proposition 17(b)), we have $\|\mathbf{s}_k\| \ge \|\mathbf{s}_1\| = \frac{\|\mathbf{r}_0\|}{\lambda_{\max}(\mathbf{A})}$ for any $k \ge 1$. Thus, we obtain that $\|\mathbf{r}_k\|_2 \le \alpha \|\mathbf{r}_0\|_2 / \lambda_{\max}(\mathbf{A}) \le \alpha \|\mathbf{s}_k\|_2$, which completes the proof. ■

In the following lemma, we upper bound the total number of matrix-product evaluations during one execution of Subroutine 3.

Lemma 19 *When Subroutine 3 returns, the total number of matrix-vector product evaluations can be bounded by* $2\sqrt{\frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}} \log\left(\frac{2\lambda_{\max}(\mathbf{A})}{\alpha}\right)$.

Proof Combining Proposition 17 and Lemma 18, we obtain that $\|\mathbf{r}_k\|_2 \le \alpha \|\mathbf{s}_k\|_2$ if

$$2\left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1}\right)^k \le \frac{\alpha}{\lambda_{\max}(\mathbf{A})} \quad \Leftrightarrow \quad k \ge \frac{\log\left(\frac{2\lambda_{\max}(\mathbf{A})}{\alpha}\right)}{\log\left(\frac{\sqrt{\kappa(\mathbf{A})}+1}{\sqrt{\kappa(\mathbf{A})}-1}\right)}.$$

Since $\log(x) \ge (x-1)/x$ for all $x > 0$, we further have $\log\left(\frac{\sqrt{\kappa(\mathbf{A})}+1}{\sqrt{\kappa(\mathbf{A})}-1}\right) \ge \frac{2}{\sqrt{\kappa(\mathbf{A})}+1} \ge \frac{1}{\sqrt{\kappa(\mathbf{A})}}$. This completes the proof. ■

Subroutine 4 $\mathsf{ExtEvec}(\mathbf{W}; \delta, q)$

```
Input: $\mathbf{W} \in \mathbb{S}^d$, $\delta > 0$, $q \in (0,1)$
Initialize: sample $\mathbf{v}_1 \in \mathbb{R}^d$ uniformly from the unit sphere, $\beta_1 \leftarrow 0$, $\mathbf{v}_0 \leftarrow 0$
Set $\epsilon \leftarrow \frac{\delta}{2(1+\delta)}$ and the number of iterations $N \leftarrow \min\left\{\left\lceil \frac{1}{4}\epsilon^{-1/2}\log\frac{11d}{q^2} + \frac{1}{2}\right\rceil, d\right\}$
for $k = 1, \dots, N$ do
    Set $\mathbf{w}_k \leftarrow \mathbf{W}\mathbf{v}_k - \beta_k \mathbf{v}_{k-1}$
    Set $\alpha_k \leftarrow \langle \mathbf{w}_k, \mathbf{v}_k \rangle$ and $\mathbf{w}_k \leftarrow \mathbf{w}_k - \alpha_k \mathbf{v}_k$
    Set $\beta_{k+1} \leftarrow \|\mathbf{w}_k\|$ and $\mathbf{v}_{k+1} \leftarrow \mathbf{w}_k / \beta_{k+1}$
end for                                                                          } Lanczos method
Form a tridiagonal matrix $\mathbf{T} \leftarrow \mathsf{tridiag}(\beta_{2:N}, \alpha_{1:N}, \beta_{2:N})$
# Use the tridiagonal structure to compute eigenvectors of $\mathbf{T}$
Compute $(\hat{\lambda}_1, \mathbf{z}^{(1)}) \leftarrow \mathsf{MaxEvec}(\mathbf{T})$ and $(\hat{\lambda}_d, \mathbf{z}^{(d)}) \leftarrow \mathsf{MinEvec}(\mathbf{T})$
Set $\mathbf{u}^{(1)} \leftarrow \sum_{k=1}^{N} z_k^{(1)} \mathbf{v}_k$ and $\mathbf{u}^{(d)} \leftarrow \sum_{k=1}^{N} z_k^{(d)} \mathbf{v}_k$
Set $\gamma \leftarrow \max\{\hat{\lambda}_1, -\hat{\lambda}_d\}$
if $\gamma \leq 1$ then
    Return $\gamma$ and $\mathbf{S} = 0$    # Case I: $\gamma \leq 1$, which implies $\|\mathbf{W}\|_{\mathrm{op}} \leq 1 + \delta$
else if $\hat{\lambda}_1 \geq -\hat{\lambda}_d$ then
    Return $\gamma$ and $\mathbf{S} = \mathbf{u}^{(1)}(\mathbf{u}^{(1)})^\top$    # Case II: $\gamma > 1$ and $\mathbf{S}$ defines a separating hyperplane
else
    Return $\gamma$ and $\mathbf{S} = -\mathbf{u}^{(d)}(\mathbf{u}^{(d)})^\top$    # Case II: $\gamma > 1$ and $\mathbf{S}$ defines a separating hyperplane
end if
```

C.2. Implementation of ExtEvec Oracle

In this section, we describe an efficient implementation of the ExtEvec oracle in Definition 9. As we discussed in Section 3.2.2, it is closely related to the problem of computing the extreme eigenvalues and eigenvectors of a given matrix, and thus we build our method on the classical Lanczos method with a random start, where the initial vector is chosen randomly and uniformly from the unit sphere (see, e.g., (Saad, 2011; Yurtsever et al., 2021)). On a high level, given the input matrix $\mathbf{W} \in \mathbb{S}^d$, we first run the Lanczos method for a sufficiently large number of iterations to obtain $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(d)}$ as the approximation of the largest and smallest eigenvector of $\mathbf{W}$, respectively. We further define $\hat{\lambda}_1 = \langle \mathbf{W}\mathbf{u}^{(1)}, \mathbf{u}^{(1)} \rangle$ and $\hat{\lambda}_d = \langle \mathbf{W}\mathbf{u}^{(d)}, \mathbf{u}^{(d)} \rangle$ as an approximation of the largest and smallest eigenvalues of $\mathbf{W}$, and let $\gamma = \max\{\hat{\lambda}_1, -\hat{\lambda}_d\}$. To construct the output $(\gamma, \mathbf{S})$ satisfying the conditions in Definition 9, we distinguish two cases depending on γ. If $\gamma \leq 1$, then we are in **Case I**, where we return γ and $\mathbf{S} = 0$. Otherwise, if $\gamma > 1$, then we are in **Case II**, where we return γ and the rank-one matrix $\mathbf{S}$ given by

$$\mathbf{S} = \begin{cases} \mathbf{u}^{(1)}(\mathbf{u}^{(1)})^\top, & \text{if } \hat{\lambda}_1 \geq -\hat{\lambda}_d; \\ -\mathbf{u}^{(d)}(\mathbf{u}^{(d)})^\top, & \text{otherwise.} \end{cases}$$

For completeness, the full algorithm is shown in Subroutine 4.

As we will show in Lemma 21, to satisfy the conditions in Definition 9, it is sufficient to run the Lanczos method for $\mathcal{O}(\sqrt{1 + 1/\delta}\log(d/q^2))$ iterations. To prove this, we first recall a classical result in Kuczyński and Woźniakowski (1992) on the convergence behavior of the Lanczos method.

Proposition 20 ((Kuczyński and Woźniakowski, 1992, Theorem 4.2)) *Consider a symmetric matrix $\mathbf{W}$ and let $\lambda_1(\mathbf{W})$ and $\lambda_d(\mathbf{W})$ denote its largest and smallest eigenvalues, respectively. Then* *after k iterations of the Lanczos method with a random start, we find unit vectors $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(d)}$ such that*

$$\mathbb{P}(\langle \mathbf{W}\mathbf{u}^{(1)}, \mathbf{u}^{(1)}\rangle \leq \lambda_1(\mathbf{W}) - \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W}))) \leq 1.648\sqrt{d}e^{-\sqrt{\epsilon}(2k-1)},$$
$$\mathbb{P}(\langle \mathbf{W}\mathbf{u}^{(d)}, \mathbf{u}^{(d)}\rangle \geq \lambda_d(\mathbf{W}) + \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W}))) \leq 1.648\sqrt{d}e^{-\sqrt{\epsilon}(2k-1)},$$

As a corollary, to ensure that, with probability at least $1-q$,

$$\langle \mathbf{W}\mathbf{u}^{(1)}, \mathbf{u}^{(1)}\rangle > \lambda_1(\mathbf{W}) - \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})) \textit{ and } \langle \mathbf{W}\mathbf{u}^{(d)}, \mathbf{u}^{(d)}\rangle < \lambda_n(\mathbf{W}) + \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})),$$

the number of iterations can be bounded by $\lceil \frac{1}{4}\epsilon^{-1/2}\log(11d/q^2) + \frac{1}{2}\rceil$.

Lemma 21 *Let γ and $\mathbf{S}$ be the output of $\mathsf{ExtEvec}(\mathbf{W}; \delta, q)$ in Subroutine 4 after $\left\lceil \frac{1}{4}\epsilon^{-1/2}\log\frac{11d}{q^2} + \frac{1}{2}\right\rceil$ iterations. Then with probability at least $1-q$, they satisfy one of the following properties:*

- *Case I: $\gamma \leq 1$, then we have $\|\mathbf{W}\|_{\mathrm{op}} \leq 1+\delta$;*
- *Case II: $\gamma > 1$, then we have $\|\mathbf{W}/\gamma\|_{\mathrm{op}} \leq 1+\delta$, $\|\mathbf{S}\|_F = 1$ and $\langle \mathbf{S}, \mathbf{W} - \hat{\mathbf{B}}\rangle \geq \gamma - 1$ for any $\hat{\mathbf{B}}$ such that $\|\hat{\mathbf{B}}\|_{\mathrm{op}} \leq 1$.*

Proof Note that in Subroutine 4, we run the Lanczos method for $\left\lceil \frac{1}{4}\epsilon^{-1/2}\log\frac{11d}{q^2} + \frac{1}{2}\right\rceil$ iterations, where $\epsilon = \frac{\delta}{2(1+\delta)}$. Thus, by Proposition 20, with probability at least $1-q$ we have

$$\hat{\lambda}_1 \triangleq \langle \mathbf{W}\mathbf{u}^{(1)}, \mathbf{u}^{(1)}\rangle \geq \lambda_1(\mathbf{W}) - \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})), \tag{47}$$
$$\hat{\lambda}_d \triangleq \langle \mathbf{W}\mathbf{u}^{(d)}, \mathbf{u}^{(d)}\rangle \leq \lambda_d(\mathbf{W}) + \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})). \tag{48}$$

Combining (47) and (48), we get

$$(1-2\epsilon)\,(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})) \leq \hat{\lambda}_1 - \hat{\lambda}_d \quad \Rightarrow \quad \lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W}) \leq \frac{1}{1-2\epsilon}(\hat{\lambda}_1 - \hat{\lambda}_d).$$

By plugging the above inequality back into (47) and (48), we further have

$$\lambda_1(\mathbf{W}) \leq \hat{\lambda}_1 + \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})) \leq \hat{\lambda}_1 + \frac{\epsilon}{1-2\epsilon}(\hat{\lambda}_1 - \hat{\lambda}_d), \tag{49}$$
$$\lambda_d(\mathbf{W}) \geq \hat{\lambda}_d - \epsilon(\lambda_1(\mathbf{W}) - \lambda_d(\mathbf{W})) \geq \hat{\lambda}_d - \frac{\epsilon}{1-2\epsilon}(\hat{\lambda}_1 - \hat{\lambda}_d). \tag{50}$$

Recall that $\gamma = \max\{\hat{\lambda}_1, -\hat{\lambda}_d\}$. By (49) and (50), we can further bound the eigenvalues of $\mathbf{W}$ by

$$\lambda_1(\mathbf{W}) \leq \gamma + \frac{\epsilon}{1-2\epsilon}\cdot 2\gamma = \frac{\gamma}{1-2\epsilon} \quad \text{and} \quad \lambda_d(\mathbf{W}) \geq -\gamma - \frac{\epsilon}{1-2\epsilon}\cdot 2\gamma = -\frac{\gamma}{1-2\epsilon}. \tag{51}$$

Hence, we can see that $\|\mathbf{W}\|_{\mathrm{op}} = \max\{\lambda_1(\mathbf{W}), -\lambda_d(\mathbf{W})\} \leq \gamma/(1-2\epsilon) = (1+\delta)\gamma$. Now we distinguish three cases.

(a) If $\gamma \leq 1$, then we are in **Case I** and the $\mathsf{ExtEvec}$ oracle outputs γ and $\mathbf{S} = 0$. In this case, we indeed have $\|\mathbf{W}\|_{\mathrm{op}} \leq (1+\delta)\gamma \leq 1+\delta$.

(b) If $\gamma > 1$ and $\hat{\lambda}_1 \geq -\hat{\lambda}_d$, then we are in **Case II** and the ExtEvec oracle returns γ and $\mathbf{S} = \mathbf{u}^{(1)}(\mathbf{u}^{(1)})^\top$. In this case, since $\|\mathbf{W}\|_{\mathrm{op}} \leq \gamma(1+\delta)$, we have $\|\mathbf{W}/\gamma\|_{\mathrm{op}} \leq 1+\delta$. Also, since $\mathbf{u}_1$ is a unit vector, we have $\|\mathbf{S}\|_F = \|\mathbf{u}_1\| = 1$. Finally, for any $\hat{\mathbf{B}}$ such that $\|\hat{\mathbf{B}}\|_{\mathrm{op}} \leq 1$, we have

$$\langle \mathbf{S}, \mathbf{W} - \hat{\mathbf{B}} \rangle = \mathbf{u}_1^\top \mathbf{W} \mathbf{u}_1 - \mathbf{u}_1^\top \hat{\mathbf{B}} \mathbf{u}_1 \geq \hat{\lambda}_1 - 1 = \gamma - 1.$$

(c) If $\gamma > 1$ and $-\hat{\lambda}_d \geq \hat{\lambda}_1$, then we are also in **Case II** and the ExtEvec oracle returns γ and $\mathbf{S} = -\mathbf{u}^{(d)}(\mathbf{u}^{(d)})^\top$. The rest follows similarly as the case above.

This completes the proof. ■

C.3. Proof of Theorem 14

Now that we have specified the implementation details of the LinearSolver and ExtEvec oracles, we move to the proof of Theorem 14. We divide the proof into the following three lemmas, which address the number of gradient evaluations, the number of matrix-vector products in LinearSolver, and the number of matrix-vector products in ExtEvec, respectively. In the following, we present the general case for any $\alpha_2 \in (0,1)$.

Lemma 22 *If we run Algorithm 1 as specified in Theorem 10 for N iterations, then the total number of line search steps can be bounded by $2N + \log_{1/\beta}(\sigma_0 L_1/\alpha_2)$.*

Proof Let l_k denote the number of line search steps in iteration k. We first note that $\eta_k = \sigma_k \beta^{l_k - 1}$ by our line search subroutine, which implies $l_k = \log_{1/\beta}(\sigma_k/\eta_k) + 1$. Thus, the total number of line search steps after N iterations can be bounded by

$$\begin{aligned}
\sum_{k=0}^{N-1} l_k = \sum_{k=0}^{N-1} \left(\log_{1/\beta} \frac{\sigma_k}{\eta_k} + 1 \right) &= N + \log_{1/\beta} \frac{\sigma_0}{\eta_0} + \sum_{k=1}^{N-1} \log_{1/\beta} \frac{\sigma_k}{\eta_k} \\
&= N + \log_{1/\beta} \frac{\sigma_0}{\eta_0} + \sum_{k=1}^{N-1} \log_{1/\beta} \frac{\eta_{k-1}}{\beta \eta_k} \qquad (52) \\
&= 2N - 1 + \log_{1/\beta} \frac{\sigma_0}{\eta_0} + \sum_{k=1}^{N-1} \log_{1/\beta} \frac{\eta_{k-1}}{\eta_k} \\
&= 2N - 1 + \log_{1/\beta} \frac{\sigma_0}{\eta_{N-1}}, \qquad (53)
\end{aligned}$$

where we used the fact that $\sigma_k = \eta_{k-1}/\beta$ for $k \geq 1$ in (52). Since we have $\eta_{N-1} \geq \alpha_2 \beta / L_1$ by Lemma 11, the lemma follows immediately from (53). ■

Note that each line search step consists of one gradient evaluation. Additionally, in each iteration of Algorithm 1, we also need to query the gradient at $\mathbf{x}_k$. Thus, as a corollary of Lemma 22, we conclude that the total number of gradient evaluations is bounded by $3N + \log_{1/\beta}(\sigma_0 L_1/\alpha_2)$.

Lemma 23 *If we run Algorithm 1 as specified in Theorem 10 for N iterations, then the total number of matrix-vector products in* ExtEvec *can be bounded by*

$$N_\epsilon \Big(\frac{1}{4} \sqrt{\frac{2L_1}{\mu}} \log \frac{70 d N_\epsilon^2 \log^4(N_\epsilon)}{p^2} + \frac{3}{2} \Big).$$

Proof It directly follows from Lemma 21 and our choice of parameters, where $\delta = \min\{\frac{\mu}{L_1-\mu}, 1\}$, $\epsilon = \frac{\delta}{2(1+\delta)} \geq \frac{\mu}{2L_1}$ and $q_t = {}^{p}\!/_{2.5(t+1)\log^2(t+1)} \leq {}^{p}\!/_{2.5N_\epsilon \log^2(N_\epsilon)}$. ■

Lemma 24 *Let N_ϵ be the minimum number of iterations required by Algorithm 1 to achieve $\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \epsilon$. Then the total number of matrix-vector products in* $\mathsf{LinearSolver}$ *can be bounded by*

$$2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2L_1\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\alpha_1\beta\mu\epsilon}\right) \cdot \left(2N_\epsilon + \log_{1/\beta}\frac{\sigma_0 L_1}{\alpha_2}\right) + 2\sqrt{\frac{3L_1}{\mu}} C_0,$$

where $C_0 \triangleq \log\left(\frac{2}{\alpha_1}\left(1+\frac{3}{2}\sigma_0 L_1\right)\right)\left(\log_{1/\beta}\left(\frac{\sigma_0 L_1}{\alpha_2\beta}\right)+1\right)$ is an constant depending on the hyper-parameters.

Proof Consider the k-th iteration. Note that in each call of $\mathsf{LinearSolver}$ in Subroutine 1, the input matrix $\mathbf{A}$ is given by $\mathbf{A} = \mathbf{I} + \eta_+\mathbf{B}_k$ with $\eta_+ \leq \sigma_k$. Therefore, we can bound $\frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})} = \frac{1+\eta_+\lambda_{\max}(\mathbf{B}_k)}{1+\eta_+\lambda_{\min}(\mathbf{B}_k)} \leq \frac{\lambda_{\max}(\mathbf{B}_k)}{\lambda_{\min}(\mathbf{B}_k)}$. Moreover, since $\frac{\mu}{2}\mathbf{I} \preceq \mathbf{B}_k \preceq (L_1 + \frac{\mu}{2})\mathbf{I}$, we have $\frac{\lambda_{\max}(\mathbf{B}_k)}{\lambda_{\min}(\mathbf{B}_k)} \leq \frac{2L_1+\mu}{\mu} \leq \frac{3L_1}{\mu}$. Hence, by Lemma 19, the number of matrix-vector product evaluations in each call of $\mathsf{LinearSolver}$ can be bounded by

$$\mathsf{MV}_k \leq 2\sqrt{\frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}} \log\left(\frac{2\lambda_{\max}(\mathbf{A})}{\alpha_1}\right) \leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2}{\alpha_1}\left(1+\sigma_k\left(L_1+\frac{\mu}{2}\right)\right)\right).$$

Moreover, since we have $\sigma_k = \eta_{k-1}/\beta$ for $k \geq 1$, we further get

$$\begin{aligned}\mathsf{MV}_k &\leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2}{\alpha_1}\left(1+\frac{\eta_{k-1}L_1}{\beta}+\frac{\eta_{k-1}\mu}{2\beta}\right)\right)\\ &\leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2L_1}{\alpha_1\beta\mu}(1+2\eta_{k-1}\mu)\right)\\ &\leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2L_1}{\alpha_1\beta\mu}\right) + 2\sqrt{\frac{3L_1}{\mu}} \log(1+2\eta_{k-1}\mu).\end{aligned}$$

Let l_k denote the number of line search steps in iteration k, and then we can bound the total number of matrix-vector products by $\sum_{k=0}^{N_\epsilon-1} l_k \cdot \mathsf{MV}_k$. Moreover, from the proof of Lemma 22, we know that $l_k = \log_{1/\beta}(\sigma_k/\eta_k) + 1$. For $k = 0$, we have

$$l_0 \leq \log_{1/\beta}\left(\frac{\sigma_0}{\eta_0}\right) + 1 \leq \log_{1/\beta}\left(\frac{\sigma_0 L_1}{\alpha_2\beta}\right) + 1 \quad \text{and} \quad \mathsf{MV}_0 \leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2}{\alpha_1}\left(1+\frac{3}{2}\sigma_0 L_1\right)\right),$$

where we used the fact that $\eta_0 > \frac{\alpha_2\beta}{L_1}$ by Lemma 11. On the other hand, we first show that

$$\prod_{k=0}^{N_\epsilon-2}(1+2\eta_k\mu) \leq \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\epsilon}. \tag{54}$$

To see this, note that by Proposition 1, it holds that $\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_0 - \mathbf{x}^*\|^2 \prod_{k=0}^{N-1}(1+2\eta_k\mu)^{-1}$. Then (54) follows from the fact that N_ϵ is the minimum number of iterations to achieve $\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \epsilon$. Thus, we have

$$\begin{aligned}\sum_{k=1}^{N-1} l_k \cdot \mathsf{MV}_k &\leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2L_1}{\alpha_1\beta\mu}\right)\sum_{k=1}^{N-1} l_k + 2\sqrt{\frac{3L_1}{\mu}}\sum_{k=1}^{N-1}\log(1+2\eta_{k-1}\mu)\cdot l_k \\ &\leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2L_1}{\alpha_1\beta\mu}\right)\sum_{k=1}^{N-1} l_k + 2\sqrt{\frac{3L_1}{\mu}}\sum_{k=1}^{N-1}\log(1+2\eta_{k-1}\mu)\cdot \sum_{k=1}^{N-1} l_k \\ &\leq 2\sqrt{\frac{3L_1}{\mu}} \log\left(\frac{2L_1\|\mathbf{x}_0-\mathbf{x}^*\|^2}{\alpha_1\beta\mu\epsilon}\right)\cdot\left(2N + \log_{1/\beta}\frac{\sigma_0 L_1}{\alpha_2}\right),\end{aligned}$$

where we used (54) and Lemma 22 in the last inequality. The proof is complete. ■

Appendix D. Additional Discussions

D.1. The Cost of Euclidean Projection

Recall that in our learning problem in Section 3.2, the action set is given by $\mathcal{Z}' \triangleq \{\mathbf{B} \in \mathbb{S}_+^d : \frac{\mu}{2}\mathbf{I} \preceq \mathbf{B} \preceq (L_1 + \frac{\mu}{2})\mathbf{I}\}$. The Euclidean projection on this set has a closed form solution. Specifically, for a given matrix $\mathbf{A} \in \mathbb{S}^d$, we first compute its eigendecomposition $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where $\mathbf{V}$ is an orthogonal matrix and $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_d)$ is a diagonal matrix. Then the Euclidean projection of $\mathbf{A}$ onto $\mathcal{Z}'$ is given by $\mathbf{V}\mathbf{\Lambda}'\mathbf{V}^\top$, where $\mathbf{\Lambda}'$ is a diagonal matrix with the diagonals being $\lambda'_k = \min\{L_1 + \frac{\mu}{2}, \max\{\frac{\mu}{2}, \lambda_k\}\}$ for $1 \leq k \leq d$. However, note that the complexity of computing the eigendecomposition is $\mathcal{O}(d^3)$, which could be prohibitive in practice. On the contrary, our projection-free online learning algorithm relies on the $\mathsf{ExtEvec}$ oracle, which can be implemented by using matrix-vector products as we detailed in Section C.2.

D.2. Complexity Bound

In this section, we derive the complexity bound of QNPE from Theorem 10. Specifically, Let N_ϵ be the minimum number of iterations required by QNPE to achieve $\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \epsilon$, and our goal is to upper bound N_ϵ in terms of the accuracy tolerance ϵ. From the linear convergence result in Theorem 10, we have

$$\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_0 - \mathbf{x}^*\|^2\left(1+\frac{\mu}{4L_1}\right)^{-N}, \tag{55}$$

and also from the superlinear convergence result we have

$$\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_0 - \mathbf{x}^*\|^2\left(1+\frac{\mu}{4L_1}\sqrt{\frac{N}{N_{\mathrm{tr}}}}\right)^{-N}, \tag{56}$$

where N_{tr} is defined as $N_{\mathrm{tr}} \triangleq \frac{4}{3} + \frac{48}{L_1^2}\|\mathbf{B}_0 - \nabla^2 f(\mathbf{x}^*)\|_F^2 + \left(\frac{36}{L_1^2} + \frac{64}{3\mu L_1}\right)L_2^2\|\mathbf{x}_0 - \mathbf{x}^*\|^2$.

From (55), to ensure that $\|\mathbf{x}_N - \mathbf{x}^*\|^2 \leq \epsilon$, it is sufficient to have

$$\|\mathbf{x}_0 - \mathbf{x}^*\|^2\left(1+\frac{\mu}{4L_1}\right)^{-N} \leq \epsilon \quad \Leftrightarrow \quad N \geq \frac{1}{\log(1+\frac{\mu}{4L_1})}\log\frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\epsilon}. \tag{57}$$

On the other hand, from (56), it is sufficient to have

$$\|\mathbf{x}_0 - \mathbf{x}^*\|^2 \left(1 + \frac{\mu}{4L_1}\sqrt{\frac{N}{N_{\text{tr}}}}\right)^{-N} \leq \epsilon. \tag{58}$$

To derive a bound on N from (58), we let N^* be the number such that the inequality above becomes equality. Then (58) holds for all $N \geq N^*$. By using the elementary inequality $log(1+x) \leq x$ for $x \geq -1$, we have

$$\log \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\epsilon} = N^* \log \left(1 + \frac{\mu}{4L_1}\sqrt{\frac{N^*}{N_{\text{tr}}}}\right) \leq \frac{\mu}{4L_1\sqrt{N_{\text{tr}}}}(N^*)^{3/2},$$

which implies that

$$N^* \geq \left(\frac{4L_1\sqrt{N_{\text{tr}}}}{\mu}\log\frac{1}{\epsilon}\right)^{2/3}.$$

Furthermore, we have

$$\log \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\epsilon} = N^* \log \left(1 + \frac{\mu}{4L_1}\sqrt{\frac{N^*}{N_{\text{tr}}}}\right) \geq N^* \log\left(1 + \left(\frac{\mu^2}{16L_1^2 N_{\text{tr}}}\log\frac{1}{\epsilon}\right)^{1/3}\right),$$

which implies

$$N^* \leq \frac{1}{\log\left(1 + \left(\frac{\mu^2}{16L_1^2 N_{\text{tr}}}\log\frac{1}{\epsilon}\right)^{1/3}\right)} \log\left(\frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\epsilon}\right). \tag{59}$$

Thus, by combining (57) and (59), we obtain

$$N_\epsilon \leq \min\left\{\frac{1}{\log(1+\frac{\mu}{4L_1})}, \frac{1}{\log\left(1 + \left(\frac{\mu^2}{16L_1^2 N_{\text{tr}}}\log\frac{1}{\epsilon}\right)^{1/3}\right)}\right\} \log\frac{\|\mathbf{x}_0 - \mathbf{x}^*\|^2}{\epsilon}.$$

Appendix E. Experimental Details

In this section, we provide more details on the dataset generation process and the implementation of gradient descent, BFGS and our proposed QNPE algorithm.

Dataset generation. We first randomly generate the underlying true feature vectors $\mathbf{a}_1^*, \ldots, \mathbf{a}_n^* \in \mathbb{R}^{d-1}$ and the underlying true parameter $\mathbf{x}^* \in \mathbb{R}^{d-1}$. Specifically, each entry of $\{\mathbf{a}_i^*\}_{i=1}^n$ and $\mathbf{x}^*$ is drawn independently according to the standard normal distribution $\mathcal{N}(0,1)$. Then the i-th feature vector $\mathbf{a}_i$ and the corresponding label y_i are given by

$$\mathbf{a}_i = \begin{bmatrix}\mathbf{a}_i^* + \mathbf{n}_i + \mathbf{1} \\ 1\end{bmatrix} \in \mathbb{R}^d \quad \text{and} \quad y_i = \text{sign}(\langle \mathbf{a}_i^*, \mathbf{x}^*\rangle) \in \{-1, +1\},$$

where $\mathbf{n}_i \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ is the i.i.d. Gaussian noise vector and $\mathbf{1}$ denotes the all-one vector. In the experiment we set $\sigma = 0.8$.

Gradient descent. The update rule is given by $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \nabla f(\mathbf{x}_k)$, where the step size η_k is selected by a backtracking line search scheme. Specifically, we choose η_k to be the largest step size in the set $\{\sigma_k \beta^i : i \geq 0\}$ that guarantees a sufficient decrease in the function value:

$$f(\mathbf{x}_k - \eta_k \nabla f(\mathbf{x}_k)) \leq f(\mathbf{x}_k) - \frac{\eta_k}{2} \|\nabla f(\mathbf{x}_k)\|^2.$$

Moreover, we set $\eta_{k+1} = \eta_k / \beta$ for $k \geq 0$ similar to the strategy in Algorithm 1. In the experiment, we set $\beta = 0.5$.

BFGS. We implemented the classical BFGS algorithm, where we employ the Moré-Thuente line search scheme using the code by O'Leary (1991). In the experiment, we set the initial Hessian approximation matrix as $\mathbf{B}_0 = L_1 \mathbf{I}$.

Our proposed QNPE algorithm. In the experiments, we set the line search parameters in Subroutine 1 by $\alpha_1 = \alpha_2 = \beta = 0.5$. We also set $\mathbf{B}_0 = \mu \mathbf{I}$ and $\rho = 1$ in Subroutine 2.