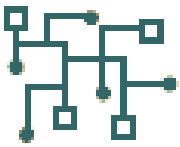


ISE

Industrial and Systems Engineering

Proximal Quasi-Newton Methods for Convex Optimization

Hiva Ghanbari

Department of Industrial and Systems Engineering
Lehigh University, Bethlehem, PA, USA

Katya Scheinberg

Department of Industrial and Systems Engineering
Lehigh University, Bethlehem, PA, USA

COR@L Technical Report 16T-06

LEHIGH UNIVERSITY

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Proximal Quasi-Newton Methods for Convex Optimization

Hiva Ghanbari and Katya Scheinberg

July 11, 2016

Abstract In [19], a general, inexact, efficient proximal quasi-Newton algorithm for composite optimization problems has been proposed and a sublinear global convergence rate has been established. In this paper, we analyze the convergence properties of this method, both in the exact and inexact setting, in the case when the objective function is strongly convex. We also investigate a practical variant of this method by establishing a simple stopping criterion for the subproblem optimization. Furthermore, we consider an accelerated variant, based on FISTA [1], to the proximal quasi-Newton algorithm. A similar accelerated method has been considered in [7], where the convergence rate analysis relies on very strong impractical assumptions. We present a modified analysis while relaxing these assumptions and perform a practical comparison of the accelerated proximal quasi-Newton algorithm and the regular one. Our analysis and computational results show that acceleration may not bring any benefit in the quasi-Newton setting.

Keywords Convex composite optimization, Strong convexity, Proximal quasi-Newton methods, Inexact conditions, Nesterov's accelerated scheme, Convergence rates, Randomized coordinate descent.

1 Introduction

We address the following *convex composite optimization* problem

$$\min_x \{F(x) := f(x) + g(x),\ x \in \mathbb{R}^n\}, \tag{1.1}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ is a continuous convex function which is possibly nonsmooth and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex continuously differentiable function with Lipschitz continuous gradient, i.e.

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \text{for every} \quad x, y \in \mathbb{R}^n,$$

where L is the global Lipschitz constant of the gradient ∇f. This class of problems, when $g(x) = \lambda\|x\|_1$, contains some of the most common machine learning models such as sparse logistic regression [27,22], sparse inverse covariance selection [6,15,18], and unconstrained Lasso [24].

The *Proximal Gradient Algorithm (PGA)* is a variant of the *proximal methods* and is a well-known first-order method for solving optimization problem (1.1). Although classical *subgradient methods* can be applied to problem (1.1) when g is nonsmooth, they can only achieve the rate of convergence of $\mathcal{O}(1/\sqrt{k})$ [11], while *PGA* converges at a rate of $\mathcal{O}(1/k)$ in both smooth and nonsmooth cases [13,1]. In order to improve the global sublinear rate of convergence of *PGA* further, the *Accelerated Proximal Gradient Algorithm (APGA)* has been originally proposed

Katya Scheinberg
Dept. of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA.
E-mail: katyascheinberg@gmail.com

Hiva Ghanbari
Dept. of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA.
E-mail: hiva.ghanbari@lehigh.edu

by Nesterov in [10], and refined by Beck and Teboulle in [1]. It has been shown that the *APGA* provides a significant improvement compared to *PGA*, both theoretically, with a rate of convergence of $\mathcal{O}(1/k^2)$, and practically [1]. This rate of convergence is known to be the best that can be obtained using only the first-order information [9,11,17], causing *APGA* to be known as the *optimal first-order method*. The class of accelerated methods contains many variants that share the same convergence rates and use only first-order information [11, 12,25]. The main known drawback of most of the variants of *APGA* is that the sequence of the step-size $\{\mu_k\}$ has to be nonincreasing. This theoretical restriction sometimes has a big impact on the performance of this algorithm in practice. In [17], in order to overcome this difficulty, a new version of *APGA* has been proposed. This variant of *APGA* allows to increase step-sizes from one iteration to the next, but maintain the same rate of convergence of $\mathcal{O}(1/k^2)$. In particular, the authors have shown that a full backtracking strategy can be applied in *APGA* and that the resulting complexity of the algorithm depends on the average value of step-size parameters, which is closely related to local Lipschitz constants, rather than the global one.

To make *PGA* and *APGA* practical, for some complicated instances of (1.1), one needs to allow for inexact computations in the algorithmic steps. In [20], inexact variants of *PGA* and *APGA* have been analyzed with two possible sources of error: one in the calculation of the gradient of the smooth term and the other in the proximal operator with respect to the nonsmooth part. The convergence rates are preserved if the sequence of errors converges to zero sufficiently fast. Moreover, it has been shown that both of these algorithms obtain linear rate of convergence, when the smooth term f is strongly convex[1]. Recently, in [4], the linear convergence of *PGA* has been shown under the *quadratic growth condition*, which is weaker than strong convexity assumption. In particular, their analysis relies on the fact that *PGA* linearly bounds the distance to the solution set by the step lengths. This property, called *error bound condition*, has been proved to be equivalent with the standard quadratic growth condition. More precisely, strong convexity assumption is a sufficient, but not a necessary condition for this error bound property.

While *PGA* and *APGA* can be efficient in solving (1.1), it has been observed that using (partial) second-order information often significantly improves the performance of the algorithms. Hence, Newton type proximal algorithms, also known as the *proximal Newton methods*, have become popular [23,15,8,2,19] and are often the methods of choice. When accurate (or nearly accurate) second-order information is used, the method no longer falls in the first-order category and faster convergence rates are expected, at least locally. Indeed, the global convergence and the local superlinear rate of convergence of the *Proximal Quasi-Newton Algorithm (PQNA)* are presented in [8] and [2], respectively for both the exact and inexact settings. However, in the case of limited memory BFGS method [3,14], the method is still essentially a first-order method. While practical performance may be by far superior, the rates of convergence are at best the same as those of the pure first-order counterparts. In [19], an inexact *PQNA* with global sublinear rate of $\mathcal{O}(1/k)$ is proposed. While the algorithm can use any positive definite Hessian estimates, as long as their eigenvalues are uniformly bounded from above and below, the practical implementation proposed in [19] used limited memory BFGS Hessian approximation. The inexact setting of the algorithm allows for a relaxed sufficient decrease condition as well as an inexact subproblem optimization, for example via coordinate descent.

In this work, we show that *PQNA*, as proposed in [19], has the linear convergence rate in the case of strongly convex smooth term f. Moreover, we consider an inexact variant, similar to the ones in [2,19], which allows for inexact subproblem solver as well as a relaxed sufficient decrease condition. In order to control the errors in the inexact subproblem optimization, we establish a simple stopping criterion for the subproblem solver, based on the iteration count, which guarantees that the inner subproblem is solved to the required accuracy. In contrast, in related works [7,26], it is assumed that an approximate subproblem solver yields an entire approximate subdifferential, which is a strong assumption on the subproblem solver which also does not clearly result in a simple stopping criterion.

Next, we apply Nesterov's acceleration scheme to *PQNA* as proposed in [19], with a view of developing the version of this algorithm with faster convergence rates in the general convex case. In [7], the authors have introduced the *Accelerated Proximal Quasi-Newton Algorithm (APQNA)* with rate of convergence of $\mathcal{O}(1/k^2)$. However, this rate of convergence is achieved under condition $0 \prec H_k \preceq H_{k-1}$, on the Hessian estimate H_k, at each iteration k. At the same time, this sequence of matrices has to be chosen so that H_k is sufficiently positive definite to provide an overapproximation of f. Hence, these two conditions may contradict with each other unless the sequence of $\{H_k\}$ consists of unnecessarily large matrices. Moreover, in a particular case, when H_k is set to be a scalar multiple of the identity, i.e., $H_k = \frac{1}{\mu_k} I$, then assumption $0 \prec H_k \preceq H_{k-1}$ enforces $\mu_k \geq \mu_{k-1}$, implying nondecreasing step-size parameters, which contradicts the standard condition of *APGA*, which is $\mu_k \leq \mu_{k-1}$.

[1] For APGA a different variant is analyzed in the case of strong convexity

In this work, we introduce a new variant of *APQNA*, where we relax the restrictive assumptions imposed in [7]. We use the scheme, originally introduced in [17], which allows for the increasing and decreasing step-size parameters. We show that our version of *APQNA* achieves the convergence rate of $\mathcal{O}(1/k^2)$ under some assumptions on the Hessian estimates. While we show that this assumption is rather strong and may not be satisfied by general matrices, it is not contradictory. Firstly, our result applies under the same restrictive condition from [7]. We also show that our condition on the matrices holds in the case when the approximate Hessian at each iteration is a scaled version of the same "fixed" matrix H, which is a generalization of *APGA*. We investigate the performance of this algorithm in practice, and discover that this restricted version of a proximal quasi-Newton method is quite effective in practice. We also demonstrate that the general L-BFGS based *PQNA* does not benefit from the acceleration, which supports our analysis of the theoretical limitations.

This paper is organized as follows. In the next section, we describe the basic definitions and existing algorithms, *PGA*, *APGA* and *PQNA*, that we refer to later in the paper. In Section 3, we analyze *PQNA* in the strongly convex case. In Section 4, we propose, state and analyze a general *APQNA* and its simplified version. We present computational results in Section 5. Finally, we state our conclusions in Section 6.

2 Notation and Preliminaries

In this work, the Euclidean norm $\|a\|^2 := a^T a$, and the inner product $\langle a, b\rangle := a^T b$, are also defined in the scaled setting such that, $\|a\|_H^2 := a^T H a$, and $\langle a, b\rangle_H := a^T H b$. We denote the identity matrix by $I \in \mathbb{R}^{n\times n}$. The vector e_j stands for a unit vector along the j-th coordinate. Finally, we use x_k to denote the approximate minimizer (the iterate), computed at iteration k of an appropriate algorithm, and x_* to denote the exact minimizer of F.

The *proximal mapping* of a convex function g at a given point v, with parameter μ is defined as

$$prox_g^\mu(v) := \arg\min_{u\in\mathbb{R}^n} \{g(u) + \frac{1}{2\mu}\|u - v\|^2\}, \quad \text{where} \quad \mu > 0. \tag{2.1}$$

The point $prox_g^\mu(v)$ minimizes function g and simultaneously, stays as close as possible to the point v. The parameter μ represents the relative weight for this trade-off. The proximal mapping is the base operation of the *proximal methods*. In order to solve the composite problem (1.1), each iteration of *PGA* computes the proximal mapping of the function g at point $x_k - \mu_k \nabla f(x_k)$ as follows:

$$\begin{aligned} p_{\mu_k}(x_k) &:= prox_g^{\mu_k}(x_k - \mu_k \nabla f(x_k)) \\ &:= \arg\min_{u\in\mathbb{R}^n} \{g(u) + \langle \nabla f(x_k), u - x_k\rangle + \frac{1}{2\mu_k}\|u - x_k\|^2\}. \end{aligned} \tag{2.2}$$

We will call the objective function, that is minimized in (2.2), a *composite quadratic approximation* of the convex function $F(x) := f(x) + g(x)$. This approximation at a given point v, for a given μ is defined as

$$Q_\mu(u, v) := f(v) + \nabla f(v)^T(u - v) + \frac{1}{2\mu}\|u - v\|^2 + g(u). \tag{2.3}$$

Then, the proximal operator can be written as

$$p_\mu(v) := \arg\min_{u\in\mathbb{R}^n} Q_\mu(u, v).$$

Using this notation we first present the basis *PGA* framework with backtracking over μ in Algorithm 1. The simple backtracking scheme enforces that the sufficient decrease condition

$$F(x_{k+1}) \le Q_{\mu_k}(x_{k+1}, x_k) \le Q_{\mu_k}(x_k, x_k) = F(x_k), \tag{2.4}$$

holds. This condition is essential in the convergence rate analysis of *PGA* and is easily satisfied when $\mu \le 1/L$. The backtracking is used for two reasons–because constant L may not be known and because $\mu \le 1/L$ may be too pessimistic, i.e., condition (2.4) may be satisfied for much larger values of μ allowing for larger steps.

Algorithm 1 Proximal Gradient Algorithm

0. Initialize $\mu_1^0 > 0$ and $x_0 \in \mathbb{R}^n$, and choose $0 < \beta < 1$.
1. **For** $k = 1, 2, \cdots$ **do**
   (1) Choose μ_k^0 and set $\mu_k := \mu_k^0$,
   (2) Compute $p_{\mu_k}(x_k) := \arg\min_{u \in \mathbb{R}^n} Q_{\mu_k}(u, x_k)$,
   – **If** $F(p_{\mu_k}(x_k)) > Q_{\mu_k}(p_{\mu_k}(x_k), x_k)$ **then**
   - $\mu_k \leftarrow \beta \mu_k$,
   - Return to (2),
   (3) $x_{k+1} \leftarrow p_{\mu_k}(x_k)$.

We now present the accelerated variant of *PGA* stated as *APGA*, where at each iteration k, instead of constructing Q_{μ_k} at the current minimizer x_k, it is constructed at a different point y_k, which is chosen as a specific linear combination of the latest two or more minimizers, e.g.

$$y_{k+1} = x_k + \alpha_k(x_k - x_{k-1}),$$

where the sequence $\{\alpha_k\}$ is chosen in such a way to guarantee the accelerated convergence rate in compared to the original *PGA*. Algorithm 2 is a variant of *APGA* framework, often referred to as FISTA, presented in [1], where $\alpha_k = (t_k - 1)/(t_{k+1})$. In this work, we choose to focus on FISTA algorithm specifically. The choice of the accelerated parameter t_{k+1} in (2.5a) is dictated by the analysis of the complexity of FISTA [1] and the condition $\mu_{k+1} \leq \mu_k$ that is imposed by the initialization of the backtracking procedure with $\mu_{k+1}^0 := \mu_k$. In [17], the definition of t_{k+1} was generalized to allow $\mu_{k+1}^0 > \mu_k$, while retaining the convergence rate. We will use a similar technique in our proposed *APQNA*.

Algorithm 2 Accelerated Proximal Gradient Algorithm

0. Initialize $t_1 = 1$, $\mu_1^0 > 0$, and $y_1 = x_0 \in \mathbb{R}^n$, and choose $0 < \beta < 1$.
1. **For** $k = 1, 2, \cdots$ **do**
   (1) Set $\mu_k := \mu_k^0$,
   (2) Compute $p_{\mu_k}(y_k) := \arg\min_{u \in \mathbb{R}^n} Q_{\mu_k}(u, y_k)$,
   – **If** $F(p_{\mu_k}(y_k)) > Q_{\mu_k}(p_{\mu_k}(y_k), y_k)$ **then**
   - $\mu_k \leftarrow \beta \mu_k$,
   - Return to (2),
   (3) $x_k \leftarrow p_{\mu_k}(y_k)$,
   (4) Set $\mu_{k+1}^0 := \mu_k$ and compute t_{k+1} and y_{k+1}, so that

$$t_{k+1} = (1 + \sqrt{1 + 4t_k^2})/2, \tag{2.5a}$$

$$\text{and} \quad y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}). \tag{2.5b}$$

In this work, we are interested in the extensions of the above proximal methods, which utilize an approximation function Q_μ, using partial second-order information about f. These quasi-Newton type proximal algorithms use a generalized form of the proximal operator (2.1), known as the *scaled proximal mapping* of g, which are defined for a given point v as

$$prox_g^H(v) := \arg\min_{u \in \mathbb{R}^n} \{g(u) + \frac{1}{2}\|u - v\|_H^2\},$$

where matrix H is a positive definite matrix. In particular, the following operator

$$p_{H_k}(x_k) := prox_g^{H_k}(x_k - H_k^{-1}\nabla f(x_k)), \tag{2.6}$$

computes the minimizer, over u, of the following composite quadratic approximation of function F

$$Q_H(u, v) := f(v) + \langle \nabla f(v), u - v\rangle + \frac{1}{2}\|u - v\|_H^2 + g(u), \tag{2.7}$$

when $v = x_k$. Matrix H is the approximate Hessian of f and its choice plays the key role in the quality of this approximation. Clearly, when $H = \frac{1}{\mu} I$, approximation (2.7) converts to (2.3), which is used throughout *PGA*.

If we set $H = \nabla^2 f(v)$, then (2.7) is the exact second-order approximation of F. At each iteration of *PQNA* the following optimization problem needs to be solved

$$p_H(v) := \arg\min_{u\in\mathbb{R}^n} Q_H(u, v), \tag{2.8}$$

which we assume to be computationally inexpensive relative to solving (1.1) for any $v \in \mathbb{R}^n$ and some chosen class of positive definite approximate Hessian H. Our assumption is motivated by [19], where it is shown that for L-BFGS Hessian approximation, problem (2.8) can be solved efficiently and inexactly via coordinate descent method. Specifically, the proximal operator (2.6) does not have closed form solution for most types of Hessian estimates H_k and most nonsmooth terms g, such as $g = \lambda\|x\|_1$, when (2.6) is a convex quadratic optimization problem. Hence, it may be too expensive to seek the exact solution of subproblem (2.6) on every iteration. In [19], an efficient version of *PQNA* is proposed that constructs Hessian estimates based on the L-BFGS updates, resulting in H_k matrices that are sum of a diagonal and a low rank matrix. The resulting subproblem, structured as (2.6), is then solved up to some expected accuracy via randomized coordinate descent, which effectively exploits the special structure of H_k. The analysis in [19] shows that the resulting inexact *PQNA* converges sublinearly if the Hessian estimates remain positive definite and bounded. The framework of the inexact variant of *PQNA* is stated in Algorithm 3. In this algorithm, the inexact solution of (2.8) is denoted by $p_{H,\varepsilon}$, as an $\varepsilon-$minimizer of the subproblem that satisfies

$$g(p_{H,\varepsilon}(v)) + \frac{1}{2}\|p_{H,\varepsilon}(v) - z\|_H^2 \leq \min_{u\in\mathbb{R}^n}\{g(u) + \frac{1}{2}\|u - z\|_H^2\} + \varepsilon, \tag{2.9}$$

where $z := v - H^{-1}\nabla f(v)$. Moreover, for a given $\eta \in (0, 1]$, the typical condition (2.4), used in [1] and [21], is relaxed by using a trust-region like sufficient decrease condition

$$\big(F(p_{H,\varepsilon}(v)) - F(v)\big) \leq \eta\left(Q_H(p_{H,\varepsilon}(v), v) - F(v)\right). \tag{2.10}$$

This relaxed condition was proposed and tested in [19] for *PQNA* and was shown to lead to superior numerical performance, saving multiple backtracking steps during the earlier iterations of the algorithm. Note that, one can obtain the exact version of Algorithm 3 by replacing p_{H_k,ε_k} with p_{H_k}, and setting $\eta = 1$.

Algorithm 3 Inexact Proximal Quasi-Newton Algorithm

0. Initialize $x_0 \in \mathbb{R}^n$, and choose $0 < \beta < 1$ and $0 < \eta \leq 1$.
1. **For** $k = 1, 2, \cdots$ **do**
   (1) Choose $\mu_k > 0$ and bounded $G_k \succeq 0$,
   (2) Define $H_k := G_k + \frac{1}{2\mu_k} I$,
   (3) Solve subproblem $\min_{u\in\mathbb{R}^n} Q_{H_k}(u, x_k)$ to obtain the ε_k-minimizer, denoted as $p_{H_k,\varepsilon_k}(x_k)$,
   – **If** $\big(F(p_{H_k,\varepsilon_k}(x_k)) - F(x_k)\big) > \eta\left(Q_{H_k}(p_{H_k,\varepsilon_k}(x_k), x_k) - F(x_k)\right)$ **then**
   - decrease μ_k so that $\mu_k \leftarrow \beta\mu_k$,
   - Return to (2),
   (4) $x_{k+1} \leftarrow p_{H_k,\varepsilon_k}(x_k)$.

Throughout our analysis, we make the following assumptions.

Assumption 2.1

- *f is convex with Lipschitz continuos gradient with constant L,*
- *g is a lower semi-continious proper convex function,*
- *F attains its minimum at a certain $x_* \in \mathbb{R}^n$,*
- *There exists positive constants m and M such that for all $k > 0$*

$$mI \preceq H_k \preceq MI. \tag{2.11}$$

Remark 2.2 In Algorithm 3, as long as the sequence of positive definite matrices G_k has uniformly bounded eigenvalues, condition (2.11) is satisfied. In fact, since the sufficient decrease condition in Step 3 is satisfied for $H_k \succeq LI$, then it is satisfied when $\mu_k \leq 1/L$. Hence, at each iteration we have a finite and bounded number of backtracking steps and the resulting H_k has bounded eigenvalues. The lower bound on the eigenvalues of H_k is simply imposed either by choosing a positive definite G_k or bounding μ_k^0 from above.

In the next section, we analyze the convergence properties of *PQNA* when f in (1.1) is strongly convex.

3 Analysis of the Proximal Quasi-Newton Algorithm under Strongly Convexity

In this section, we analyze the convergence properties of *PQNA* to solve problem (1.1), in the case when the smooth function f is $\gamma-$strongly convex. In particular, the following assumption is made throughout this section.

Assumption 3.1 *For all x and y in $\mathbb{R}^n$, and any $t \in [0,1]$*

$$\gamma\|x-y\|^2 \leq \langle \nabla f(x) - \nabla f(y), x-y\rangle, \tag{3.1a}$$

$$\text{and} \quad f(tx+(1-t)y) \leq tf(x) + (1-t)f(y) - \frac{1}{2}\gamma t(1-t)\|x-y\|^2. \tag{3.1b}$$

To establish a linear convergence rate of *PQNA*, we consider extending two different approaches to show a similar result for *PGA*. The first approach we considered can be found in [5], and is based on the proof techniques used in [4] for *PGA*. The reason we chose the approach in [4] is due to the fact that the linear rate of convergence is shown under the quadratic growth condition, which is a relaxation of the strong convexity. Hence, extending this analysis to *PQNA*, as a subject of a future work, may allow us to relax the strong convexity assumption for this algorithm as well. However, there appears to be some limitations in the extension of this analysis [5], in particular in the inexact case. This observation motivates us to present the approach used in [13] to analyze convergence properties of inexact *PQNA*. As we see below, this analysis readily extends to our case and allows us to establish simple rules for subproblem solver termination to achieve the desired subproblem accuracy.

3.1 Convergence Analysis

Let us consider Algorithm 3 for which (2.9) holds for some sequence of errors $\varepsilon_k \geq 0$. The relaxed sufficient decrease condition

$$F(x_{k+1}) - F(x_k) \leq \eta(Q_{H_k}(x_{k+1}, x_k) - F(x_k)),$$

for a given $\eta \in (0,1]$, can be written as

$$\begin{aligned} F(x_{k+1}) &\leq Q_{H_k}(x_{k+1}, x_k) - (1-\eta)\left(Q_{H_k}(x_{k+1}, x_k) - F(x_k)\right) \\ &\leq Q_{H_k}(x_{k+1}, x_k) - \frac{1-\eta}{\eta}\left(F(x_{k+1}) - F(x_k)\right). \end{aligned}$$

Thus, at each iteration we have

$$F(x_{k+1}) \leq Q_{H_k}(x_{k+1}, x_k) + \xi_k, \tag{3.2}$$

where the sequence of the errors ξ_k is defined as

$$\xi_k \leq (1 - \frac{1}{\eta})\left(F(x_{k+1}) - F(x_k)\right). \tag{3.3}$$

In particular, setting $\eta = 1$ results in $\xi_k = 0$, for all k and enforces the algorithm to accept only those steps that achieve full (predicted) reduction. However, using $\eta < 1$ allows the algorithm to take steps satisfying only a fraction of the predicted reduction, which may lead to larger steps and faster progress.

Under the above inexact condition, we can show the following result.

Theorem 3.2 *Suppose that Assumptions 2.1 and 3.1 hold. At each iteration of the inexact PQNA, stated in Algorithm 3, we have*

$$F(x_k) - F(x_*) \leq \rho^k\left(F(x_0) - F(x_*) + A_k\right), \tag{3.4}$$

when $\rho = 1 - (\eta\gamma)/(\gamma + M)$, and

$$A_k := \eta \sum_{i=1}^{k} \left(\varepsilon_i/\rho^i\right).$$

Proof Applying (3.2), with $v = x_k$ and consequently $p_{H_k,\varepsilon_k}(x_k) = x_{k+1}$, we have

$$\begin{aligned}
F(x_{k+1}) &\leq Q_{H_k}(x_{k+1}, x_k) + \xi_k \\
&= f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{1}{2}\|x_{k+1} - x_k\|_{H_k}^2 + g(x_{k+1}) + \xi_k \\
&= \min_{u \in \mathbb{R}^n} \; f(x_k) + \langle \nabla f(x_k), u - x_k \rangle + \frac{1}{2}\|u - x_k\|_{H_k}^2 + g(u) + \varepsilon_k + \xi_k \\
&\leq \min_{u \in \mathbb{R}^n} \; f(u) + \frac{1}{2}\|u - x_k\|_{H_k}^2 + g(u) + (\varepsilon_k + \xi_k) \qquad \text{(convexity of } f\text{)} \\
&= \min_{u \in \mathbb{R}^n} \; F(u) + \frac{1}{2}\|x_k - u\|_{H_k}^2 + (\varepsilon_k + \xi_k) \\
&\leq \min_{t \in [0,1]} \; F(tx_* + (1-t)x_k) + \frac{1}{2}\|x_k - tx_* - (1-t)x_k\|_{H_k}^2 + (\varepsilon_k + \xi_k) \\
&\leq \min_{t \in [0,1]} \; tF(x_*) + (1-t)F(x_k) - \frac{1}{2}\gamma t(1-t)\|x_* - x_k\|^2 + \frac{1}{2}t^2\|x_* - x_k\|_{H_k}^2 + (\varepsilon_k + \xi_k) \qquad \text{(using (3.1b))} \\
&\leq \min_{t \in [0,1]} \; tF(x_*) + (1-t)F(x_k) - \frac{1}{2}\gamma t(1-t)\|x_* - x_k\|^2 + \frac{1}{2}Mt^2\|x_* - x_k\|^2 + (\varepsilon_k + \xi_k) \\
&\leq t'F(x_*) + (1-t')F(x_k) + (\varepsilon_k + \xi_k). \qquad \text{(where } t' = \frac{\gamma}{\gamma + M}\text{)}
\end{aligned}$$

Therefore, we have

$$F(x_{k+1}) \leq t'F(x_*) + (1-t')F(x_k) + (\varepsilon_k + \xi_k),$$

which implies

$$F(x_{k+1}) - F(x_*) \leq (1-t')(F(x_k) - F(x_*)) + (\varepsilon_k + \xi_k).$$

Now, by substituting the expression for ξ_k, as sated in (3.3), we will have

$$F(x_{k+1}) - F(x_*) \leq \rho(F(x_k) - F(x_*)) + \eta\varepsilon_k,$$

where $\rho = 1 - \eta t'$. Now, we can conclude the final result as

$$\begin{aligned}
F(x_k) - F(x_*) &\leq \rho^k(F(x_0) - F(x_*)) + \sum_{i=1}^{k} \eta\rho^{k-i}\varepsilon_i \\
&= \rho^k \left(F(x_0) - F(x_*) + \eta \sum_{i=1}^{k} \left(\varepsilon_i/\rho^i\right) \right),
\end{aligned}$$

where $\rho = 1 - (\eta\gamma)/(\gamma + M)$. □

Remark 3.3 In Theorem 3.2, by setting $\varepsilon_k = 0$ and $\eta = 1$, which implies $\xi_k = 0$, we achieve the linear convergence rate of the exact variant of *PQNA*.

Remark 3.4 We have shown that the linear rate of *PQNA* is $\rho = 1 - (\eta\gamma)/(\gamma + M)$. As argued in Remark 2.2, M is of the same order as L in the worst case, hence in that case the linear rate of *PQNA* is the same as that of the simple *PGA*. However, it is easy to see that in the proof of Theorem 3.2, the linear rate is derived using the upper bound on $\|x_* - x_k\|_{H_k}^2$, where H_k is the approximate Hessian on step k. Clearly, the idea of using the partial second-order information is to reduce the worst case bound of H_k in general and consequently on $\|x_* - x_k\|_{H_k}^2$. In particular, obtaining a smaller bound M_k on each iteration yields a larger convergence coefficient $\rho_k = 1 - (\eta\gamma)/(\gamma + M_k)$. While for general H_k, we do not expect to improve upon the regular *PGA* in theory, this remark serves to explain the much better behavior of *PQNA* in practice.

Based on the result of Theorem 3.2, it follows that the boundedness of the sequence $\{A_k\}$ is a sufficient condition to achieve the linear convergence rate. Hence, the required condition on the sequence of errors ε_i is $\sum_{i=1}^{k} \left(\varepsilon_i/\rho^i\right) < \infty$. Let us consider a sequence satisfying $\varepsilon_i \leq C\rho^{i\cdot\delta}$, for some $\delta > 1$ and some $C > 0$. Then, we have $\sum_{i=1}^{k} \left(\varepsilon_i/\rho^i\right) \leq C\sum_{i=1}^{k} \rho^{i(\delta-1)}$, which is uniformly bounded for all k. Recall that the i-th subproblem $Q_i^* := \min_{u \in \mathbb{R}^n} Q_{H_i}(u, x_i)$ is a strongly convex function with strong convexity parameter at least m–the lower bound on the eigenvalues of H_k. Let us assume now that each subproblem is solved via an algorithm with a

linear convergence rate for strongly convex problems. In particular, if the subproblem solver is applied for r iterations to the i-th subproblem, we have

$$\left(Q_{H_i}(u_r, x_i) - Q_i^*\right) \leq \alpha \left(Q_{H_i}(u_{r-1}, x_i) - Q_i^*\right), \quad \text{where} \quad \alpha \in (0,1). \tag{3.5}$$

Our goal is to ensure that $\varepsilon_i \leq C\rho^{i\cdot\delta}$, which can be achieved by applying sufficient number of iterations of the subproblem algorithm. To be specific, the following theorem characterizes this required bound on the number of inner iterations.

Theorem 3.5 *Suppose that at the i-th iteration of Algorithm 3, after applying the subproblem solver satisfying (3.5) for $r(i)$ iterations, starting with $u_0 = x_i$, we obtain solution $x_{i+1} = u_{r(i)}$. Let $r(i)$ satisfy*

$$r(i) \geq i \log_{1/\alpha}(1/\rho^{\delta}), \tag{3.6}$$

for some $\delta > 1$, and ρ defined in Theorem 3.2. Then

$$\left(Q_{H_i}(x_{i+1}, x_i) - Q_i^*\right) \leq C\rho^{i\cdot\delta},$$

holds for all i, with C being the uniform bound on $Q_{H_i}(x_i, x_i) - Q_i^*$, and the linear convergence of Algorithm 3 is achieved.

Proof First, assume that at the $i-$th iteration we have applied the subproblem solver for $r(i)$ iterations to minimize strongly convex function Q_{H_i}. Now, by combining $Q_{H_i}(u_0, u_0) - Q_i^* \leq C$ and (3.5), we can conclude the following upper bound, so that

$$Q_{H_i}(u_{r(i)}, u_0) - Q_i^* \leq \alpha^{r(i)} C.$$

Now, if $\alpha^{r(i)} C \leq \varepsilon_i$, we can guarantee that $u_{r(i)}$ is an ε_i-solution of the i-th subproblem, so that $Q_{H_i}(u_{r(i)}, u_0) \leq Q_i^* + \varepsilon_i$. Now, assuming that ρ is known, we can set the error rate of the i-th iteration as $\varepsilon_i \leq C\rho^{i\cdot\delta}$, for a fixed $\delta > 1$. In this case, the number of inner iterations which guarantees the ε_i-minimizer will be

$$r(i) \geq i \log_{1/\alpha}(1/\rho^{\delta}).$$

□

Note that, since subproblems are strongly convex, the required linear convergence rate for the subproblem solver, stated in (3.5), can be guaranteed via some basic first-order algorithms or their accelerated variants. However, one difficulty in obtaining lower bound (3.6) is that it depends on the prior knowledge of ρ. In the following, we describe how we can relax this requirement.

Remark 3.6 In Theorem 3.5, instead of condition (3.6), consider $r(i)$ satisfying

$$r(i) \geq i \log_{1/\alpha}(i/\ell), \tag{3.7}$$

for any given $\ell > 0$. Then $\varepsilon_i \leq C\,(\ell/i)^i$ implies $\varepsilon_i \leq C\rho^{i\cdot\delta}$, for sufficiently large i.

In the next subsection, we extend our analysis to the case of solving subproblems via the randomized coordinate descent, where at each iteration the error ε_i is random and satisfies the desired bound only in expectation.

3.2 Solving Subproblems via Randomized Coordinate Descent

As we mentioned before, in order to achieve linear convergence rate of the inexact *PQNA*, any simple first-order method (such as *PGA*) can be applied. However, as discussed in [19], in the case when $g(x) = \lambda\|x\|_1$ and H_k is sum of a diagonal and a low rank matrix, as in the case of L-BFGS approximations, the *coordinate descent* method is the most efficient approach to solve the strongly convex quadratic subproblems. In this case, each iteration of coordinate descent has complexity of $\mathcal{O}(m)$, where m is the memory size of L-BFGS, which is usually chosen to be less than 20, while each iteration of a proximal gradient method has complexity of $\mathcal{O}(nm)$ and each iteration of the Newton type proximal method has complexity of $\mathcal{O}(nm^2)$. While more iterations of coordinate descent may be required to achieve the same accuracy, it tends to be the most efficient approach. To extend our theory of the previous section and to establish the bound on the number of coordinate descent

Algorithm 4 Randomized Coordinate Descent Algorithm

0. **Input:** initial point $v \in \mathbb{R}^n$ and required number of iterations $r > 0$.
1. Set $\bar{u} \leftarrow v$,
2. **For** $i = 1, 2, \cdots, r$ **do**
   (1) Choose $j \in \{1, 2, \cdots, n\}$ with probability $1/n$,
   (2) Compute $z^* := \arg\min_{z \in \mathbb{R}^n} Q_H(\bar{u} + ze_j, v)$,
   (3) $\bar{u} \leftarrow \bar{u} + z^* e_j$,
3. **Output:** return $u_r \leftarrow \bar{u}$.

steps needed to solve each subproblem, we utilize convergence results for the *randomized coordinate descent* [16], as is done in [19].

Algorithm 4 shows the framework of the randomized coordinate descent method, which can be used as a subproblem solver of Algorithm 3 and is identical to the method used in [19]. In Algorithm 4, function Q_H is iteratively minimized over a randomly chosen coordinate, while the other coordinates remain fixed.

In what follows, we restate *Theorem 6* in [16], which establishes linear convergence rate of the randomized coordinate descent algorithm, in expectation, to solve strongly convex problems.

Theorem 3.7 *Suppose we apply randomized coordinate descent for r iterations, to minimize the m-strongly convex function Q with M-Lipschitz gradient, to obtain the random point u_r. When u_0 is the initial point and $Q^* := \min_{u \in \mathbb{R}^n} Q_H(u, u_0)$, for any r, we have*

$$E(Q_H(u_r, u_0) - Q^*) \leq \left(1 - \frac{1 - \phi_{m,M}}{n}\right)^r \left(Q_H(u_0, u_0) - Q^*\right), \tag{3.8}$$

where ϕ_m is defined as

$$\phi_{m,M} = \begin{cases} 1 - m/4M, & \text{if } m \leq 2M, \\ M/m, & \text{otherwise.} \end{cases} \tag{3.9}$$

Proof The proof can be found in [16]. □

Now, we want to analyze how we can utilize the result of Theorem 3.7 to achieve the linear convergence rate of inexact *PQNA*, in expectation. Toward this end, first we need the following theorem as the probabilistic extension of Theorem 3.2.

Theorem 3.8 *Suppose that Assumptions 2.1 and 3.1 hold. At each iteration i of the inexact PQNA, stated in Algorithm 3, assume that ε_i is random. Then, we have*

$$E(F(x_k) - F(x_*)) \leq \rho^k \left(F(x_0) - F(x_*) + B_k\right), \tag{3.10}$$

when $\rho = 1 - (\eta\gamma)/(\gamma + M)$, and

$$B_k := \eta \sum_{i=1}^{k} \left(E(\varepsilon_i)/\rho^i\right).$$

Proof The proof is a trivial modification of that of Theorem 3.2. □

In what follows, we describe how the randomized coordinate descent method ensures the required accuracy of subproblems and consequently guarantees linear convergence of the inexact *PQNA*.

Theorem 3.9 *Suppose that at the i-th iteration of Algorithm 3, after applying Algorithm 4 for $r(i)$ iterations, starting with $u_0 = x_i$, we obtain solution $x_{i+1} = u_{r(i)}$. If*

$$r(i) \geq i \log_{1/\alpha_n}(i/\ell),$$

where ℓ is any positive constant, $\alpha_n = \left(1 - \frac{1-\phi_{m,M}}{n}\right)$ with $\phi_{m,M}$ defined in (3.9), and C is the uniform bound on $Q_{H_i}(x_i, x_i) - Q_i^*$, then Algorithm 3, converges linearly, in expectation.

Proof Suppose that at the i-th iteration of Algorithm 3, we apply $r(i)$ steps of Algorithm 4 to minimize the strongly convex function Q_{H_i}. If $u_{r(i)}$ denotes the resulting random point, when u_0 is the initial point and $Q_i^* := \min_{u\in\mathbb{R}^n} Q_{H_i}(u, u_0)$, then based on Theorem 3.7 we have

$$E(Q_{H_i}(u_{r(i)}, u_0) - Q_i^*) \leq \alpha_n^{r(i)} C, \tag{3.11}$$

where $\alpha_n = \left(1 - \frac{1-\phi_{m,M}}{n}\right)$, with $\phi_{m,M}$ defined in (3.9), and C bounds $Q_{H_i}(u_0, u_0) - Q_i^*$. Now, based on the result of Theorem 3.8, if $E(\varepsilon_i) \leq C(\ell/i)^i$ for some given positive constant ℓ, then for sufficiently large i, we can guarantee that B_k is uniformly bounded for all k, and consequently the linear convergence rate of Algorithm 3, in expectation is established. Now, by using (3.11), $E(\varepsilon_i) \leq C(\ell/i)^i$ simply follows from

$$r(i) \geq i \log_{1/\alpha_n}(i/\ell).$$

□

Remark 3.10 The bound on the number of steps $r(i) \geq i \log_{1/\alpha_n}(i/\ell)$ for randomized coordinate descent differs from the bound $r(i) \geq i \log_{1/\alpha}(i/\ell)$ on the number of steps of a deterministic linear convergence method, such as *PGA* by the difference in constants α and α_n. It can be easily shown that in the worst case $\alpha_n \approx \alpha/n$, and hence, the number of coordinate descent steps is around n times larger than that of a proximal gradient method. On the other hand, each coordinate descent step is n times less expensive and in many practical cases a modest number of iterations of randomized coordinate descent is sufficient. Discussions on this can be found in [16] and [19] as well as in Section 5.

4 Accelerated Proximal Quasi-Newton Algorithm

We now turn to an accelerated variant of *PQNA*. As we described in the introduction section, the algorithm proposed in [7] is a proximal quasi-Newton variant of FISTA, described in Algorithm 2. In [7], the convergence rate of $\mathcal{O}(1/k^2)$ is shown under the condition that the Hessian estimates satisfy $0 \prec H_k \preceq H_{k-1}$, at each iteration. On the other hand, the sequence $\{H_k\}$ is chosen so that the quadratic approximation of f is an over approximation. This leads to an unrealistic setting where two possible contradictory conditions need to be satisfied and as mentioned earlier, this condition contradicts the assumptions of the original *APGA*, stated in Algorithm 2. We propose a more general version, henceforth referred to as *APQNA*, which allows a more general sequence of H_k and is based on the relaxed version of FISTA, proposed in [17], which does not impose monotonicity of the step-size parameters. Moreover, our algorithm allows more general Hessian estimates as we explain below.

4.1 Algorithm Description

The main framework of *APQNA* as stated in Algorithm 5 is similar to that of Algorithm 2, where the simple composite quadratic approximation Q_μ was replaced by the scaled version Q_H, as is done in Algorithm 3, using (partial) Hessian information. As in the case of Algorithm 3, we assume that the approximate Hessian H_k is a positive definite matrix such that $mI \preceq H_k \preceq MI$, for some positive constants m and M. As discussed in Remark 2.2, it is simple to show that this condition can be satisfied for any positive m and for any large enough M. Here, however, we will need additional much stronger assumptions on the sequence $\{H_k\}$. The algorithm, thus, has some additional steps compared to Algorithm 2 and the standard FISTA type proximal quasi-Newton algorithm proposed in [7]. Below, we present Algorithm 5 and discuss the steps of each iteration in detail.

The key requirement imposed by Algorithm 5 on the sequence $\{H_k\}$ is that $\sigma_{k+1}H_{k+1} \preceq \sigma_k H_k$, while $\theta_k := \sigma_k/\sigma_{k+1}$ is used to evaluate the accelerated parameter t_{k+1} through (4.1a). During Steps 4 and 5 of iteration k, initial guesses for σ_{k+1}^0 and H_{k+1} are computed and used to define θ_k, which is then used to compute t_{k+1} and y_{k+1}. Since the approximate Hessian H_{k+1} may change during Step 2 of iteration $k+1$, σ_{k+1} may need to change as well in order to satisfy condition $\sigma_{k+1}H_{k+1} \preceq \sigma_k H_k$. In particular, we may shrink the value of σ_{k+1} and consequently will need to recompute θ_k and, thus, t_{k+1} and y_{k+1}. Therefore, the backtracking process in Step 2 of Algorithm 5 involves a loop which may require repeated computations of y_k and hence $\nabla f(y_k)$.

Algorithm 5 Accelerated Proximal Quasi-Newton Algorithm

0. Initialize $t_1 = 1, \theta_0 = 1, \sigma_1^0 > 0$, $y_1 = x_{-1} = x_0 \in \mathbb{R}^n$, and positive definite matrix $H_0 \in \mathbb{R}^{n\times n}$, and choose $0 < \beta < 1$.
1. **For** $k = 1, 2, \cdots$ **do**
   (1) Set $\sigma_k := \sigma_k^0$,
   (2) Compute $p_{H_k}(y_k) := \arg\min_{u\in\mathbb{R}^n} Q_{H_k}(u, y_k)$,
   – **If** $F(p_{H_k}(y_k)) > Q_{H_k}(p_{H_k}(y_k), y_k)$ **then**
   - update $H_k \leftarrow \frac{1}{\beta} H_k$,
   - modify σ_k so that $\sigma_k H_k \preceq \sigma_{k-1} H_{k-1}$,
   - set $\theta_{k-1} = \sigma_{k-1}/\sigma_k$ and recompute t_k and y_k using (4.1a)-(4.1b),
   - Return to (2),
   (3) $x_k \leftarrow p_{H_k}(y_k)$,
   (4) Choose $\sigma_{k+1}^0 > 0$ and H_{k+1} so that $\sigma_{k+1}^0 H_{k+1} \preceq \sigma_k H_k$,
   (5) Set $\theta_k := \sigma_k/\sigma_{k+1}^0$ and compute t_{k+1} and y_{k+1}, so that

$$t_{k+1} = (1 + \sqrt{1 + 4\theta_k t_k^2})/2, \tag{4.1a}$$

$$\text{and} \quad y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}). \tag{4.1b}$$

Remark 4.1 Note that Algorithm 5 does not allow the use of exact Hessian information at y_{k+1}, i.e., $H_{k+1} = \nabla^2 f(y_{k+1})$, because it is assumed that H_{k+1} is computed before y_{k+1} (since y_{k+1} uses the value of σ_{k+1}, whose value may have to be dependent on H_{k+1}). However, it is possible to use $H_{k+1} = \nabla^2 f(x_k)$ in Algorithm 5. To use $H_{k+1} = \nabla^2 f(y_{k+1})$, one would need to be able to compute σ_{k+1} before H_{k+1} and somehow ensure that condition $\sigma_{k+1} H_{k+1} \preceq \sigma_k H_k$ is satisfied. This condition can eventually be satisfied by applying similar technique to Step 2, but in that case H_{k+1} will not be equal to the Hessian, but to some multiple of the Hessian, i.e., $\frac{1}{\beta^i}\nabla^2 f(y_{k+1})$, for some i.

One trivial choice of the matrix sequence is $H_k = \frac{1}{\mu_k} I$. In this case, the sequence of scalars $\sigma_k = \mu_k$, satisfies $\sigma_{k+1} H_{k+1} \preceq \sigma_k H_k$, for all k. This choice of Hessian reduces Algorithm 5 to the version of *APGA* with full backtracking of the step-size parameters, proposed in [17], hence Algorithm 5 is the generalization of that algorithm. Another choice for the matrix sequence is $H_k = \frac{1}{\sigma_k} H$, where the matrix H is any fixed positive definite matrix. This setting of H_k automatically satisfies condition $\sigma_{k+1} H_{k+1} \preceq \sigma_k H_k$, and Algorithm 5 reduces to the simplified version stated below in Algorithm 6.

Algorithm 6 Accelerated Proximal Quasi-Newton Algorithm with Fixed Hessian

0. Initialize $t_1 = 1, \theta_0 = 1, \sigma_1^0 > 0$, and $y_1 = x_{-1} = x_0 \in \mathbb{R}^n$, and choose positive definite matrix $H \in \mathbb{R}^{n\times n}$, and $0 < \beta < 1$.
1. **For** $k = 1, 2, \cdots$ **do**
   (1) Set $\sigma_k := \sigma_k^0$,
   (2) Set $H_k = (1/\sigma_k)H$ and compute $p_{H_k}(y_k) := \arg\min_{u\in\mathbb{R}^n} Q_{H_k}(u, y_k)$,
   – **If** $F(p_{H_k}(y_k)) > Q_{H_k}(p_{H_k}(y_k), y_k)$ **then**
   - update $\sigma_k \leftarrow \beta\sigma_k$,
   - update θ_{k-1} and recompute t_k and y_k using (4.2a)-(4.2b),
   - Return to (2),
   (3) $x_k \leftarrow p_{H_k}(y_k)$,
   (4) Choose $\sigma_{k+1}^0 > 0$ and set $\theta_k := \sigma_k/\sigma_{k+1}^0$ and compute t_{k+1} and y_{k+1}, so that

$$t_{k+1} = (1 + \sqrt{1 + 4\theta_k t_k^2})/2, \tag{4.2a}$$

$$\text{and} \quad y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}). \tag{4.2b}$$

Note that, by the same logic that was used in Remark 2.2, the number of backtracking steps at each iteration of Algorithm 6 is uniformly bounded. Thus, as long as the fixed approximate Hessian H is positive definite, a Hessian estimate $H_k = \frac{1}{\sigma_k} H$ has positive eigenvalues bounded from above and below.

In the next section, we analyze the convergence properties of Algorithm 5, where the approximate Hessian H_k is produced by some generic unspecified scheme. The motivation is to be able to apply the analysis to

popular and efficient Hessian approximation methods, such as L-BFGS. However, in the worst case for general H_k, a positive lower bound for $\{\sigma_k\}$ can not be guaranteed for such a generic scheme. This observation motivates the analysis of Algorithm 6, as a simplified version of Algorithm 5. It remains to be seen if some bound on $\{\sigma_k\}$ is possible to derive for matrices arising specifically via L-BFGS updates.

4.2 Convergence Analysis

In this section, we prove that if the sequence $\{\sigma_k\}$ is bounded away from zero, Algorithm 5 achieves the same rate of convergence as *APGA*, i.e., $\mathcal{O}(1/k^2)$. First, we state a simple result based on the optimality of p_H.

Lemma 4.2 *For any $v \in \mathbb{R}^n$, there exists a subgradient of function g where $\nu_g(p_H(v)) \in \partial g(p_H(v))$, such that*

$$\nabla f(v) + H(p_H(v) - v) + \nu_g(p_H(v)) = 0.$$

Proof The proof is followed immediately from the optimality condition of the convex optimization problem (2.8). □

Now, we can show the following lemma, which bounds the change in the objective function F and is a simple extension of a similar theorem in [1].

Lemma 4.3 *Let $v \in \mathbb{R}^n$ and $H \succ 0$ be such*

$$F(p_H(v)) \leq Q_H(p_H(v), v), \tag{4.3}$$

holds for a given v, then for any $x \in \mathbb{R}^n$

$$F(x) - F(p_H(v)) \geq \frac{1}{2}\|p_H(v) - v\|_H^2 + \langle v - x, p_H(v) - v\rangle_H.$$

Proof From (4.3), we have

$$F(x) - F(p_H(v)) \geq F(x) - Q_H(p_H(v), v). \tag{4.4}$$

Now, based on the convexity of functions f and g, we have

$$\begin{aligned} f(x) &\geq f(v) + \langle \nabla f(v), x - v\rangle, \\ \text{and} \quad g(x) &\geq g(p_H(v)) + \langle \nu_g(p_H(v)), x - p_H(v)\rangle, \end{aligned}$$

where $\nu_g(p_H(v))$ is defined in Lemma 4.2. Summing the above inequalities yields

$$F(x) \geq f(v) + \langle \nabla f(v), x - v\rangle + g(p_H(v)) + \langle \nu_g(p_H(v)), x - p_H(v)\rangle. \tag{4.5}$$

Using (2.7) and (4.5) in (4.4) yields

$$\begin{aligned} F(x) - F(p_H(v)) \geq\ & f(v) + \langle \nabla f(v), x - v\rangle + g(p_H(v)) + \langle \nu_g(p_H(v)), x - p_H(v)\rangle \\ & - f(v) - \langle \nabla f(v), p_H(v) - v\rangle - \frac{1}{2}\|p_H(v) - v\|_H^2 - g(p_H(v)) \\ =& -\frac{1}{2}\|p_H(v) - v\|_H^2 + \langle x - p_H(v), \nabla f(v) + \nu_g(p_H(v))\rangle \\ =& -\frac{1}{2}\|p_H(v) - v\|_H^2 + \langle x - p_H(v), H(v - p_H(v))\rangle \\ =& -\frac{1}{2}\|p_H(v) - v\|_H^2 + \langle x - p_H(v), H(v - p_H(v))\rangle \\ & + \langle v - p_H(v), v - p_H(v)\rangle_H - \langle v - p_H(v), v - p_H(v)\rangle_H \\ =& \frac{1}{2}\|p_H(v) - v\|_H^2 + \langle v - x, p_H(v) - v\rangle_H. \end{aligned}$$

□

The following result is a simple corollary of Lemma 4.3.

Corollary 4.4 *Let $v \in \mathbb{R}^n$ and $H \succ 0$ be such that*

$$F(p_H(v)) \leq Q_H(p_H(v), v),$$

then for any $x \in \mathbb{R}^n$

$$\begin{aligned} 2(F(x) - F(p_H(v))) &\geq \|p_H(v) - v\|_H^2 + 2\langle p_H(v) - v, v - x\rangle_H, \\ &= \|p_H(v) - x\|_H^2 - \|v - x\|_H^2. \end{aligned} \tag{4.6}$$

Proof The result immediately follows by applying the following identity

$$\|b - a\|^2 + 2(b - a)^T(a - c) = \|b - c\|^2 - \|a - c\|^2. \tag{4.7}$$

to Lemma 4.3 with

$$a := H^{\frac{1}{2}}v, \quad b := H^{\frac{1}{2}}p_H(v), \quad c := H^{\frac{1}{2}}x.$$

The next lemma states the key properties which are used in the convergence analysis.

Lemma 4.5 *At each iteration of Algorithm 5, the following relations hold*

$$\sigma_k H_k \succeq \sigma_{k+1} H_{k+1}, \tag{4.8a}$$

$$\text{and} \quad \sigma_k t_k^2 \geq \sigma_{k+1} t_{k+1}(t_{k+1} - 1). \tag{4.8b}$$

Proof The proof follows trivially from the conditions in Algorithm 5 and the fact that $\theta_k \leq \sigma_k/\sigma_{k+1}$. □

Now, using this lemma and previous results we derive the key property of the iterations of *APQNA*.

Lemma 4.6 *For all $k \geq 1$ for Algorithm 5 we have*

$$2\sigma_k t_k^2 v_k + \sigma_k u_k^T H_k u_k \geq 2\sigma_{k+1} t_{k+1}^2 v_{k+1} + \sigma_{k+1} u_{k+1}^T H_{k+1} u_{k+1},$$

where $v_k = F(x_k) - F(x_*)$ and $u_k = t_k x_k - (t_k - 1)x_{k-1} - x_*$.

Proof In (4.6), by setting $v = y_{k+1}$, $p_H(v) = x_{k+1}$, $H = H_{k+1}$, and $x = x_k$ and then by multiplying the resulting inequality by $\sigma_{k+1}(t_{k+1} - 1)$, we will have

$$\begin{aligned} 2\sigma_{k+1}(t_{k+1} - 1)(v_k - v_{k+1}) \geq\ & (t_{k+1} - 1)(x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(x_{k+1} - y_{k+1}) \\ & + 2(t_{k+1} - 1)(x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(y_{k+1} - x_k). \end{aligned}$$

On the other hand, in (4.6), by setting $x = x_*$ and multiplying it by σ_{k+1}, we have

$$\begin{aligned} -2\sigma_{k+1} v_{k+1} \geq\ & (x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(x_{k+1} - y_{k+1}) \\ & + 2(x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(y_{k+1} - x_*). \end{aligned}$$

By adding these two inequalities, we have

$$\begin{aligned} 2\sigma_{k+1}((t_{k+1} - 1)v_k - t_{k+1} v_{k+1}) \geq\ & t_{k+1}(x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(x_{k+1} - y_{k+1}) \\ & + 2(x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(t_{k+1} y_{k+1} - (t_{k+1} - 1)x_k - x_*). \end{aligned}$$

Multiplying the last inequality by t_{k+1} and applying inequality (4.8b) give

$$\begin{aligned} 2(\sigma_k t_k^2 v_k - \sigma_{k+1} t_{k+1}^2 v_{k+1}) \geq\ & t_{k+1}^2 (x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(x_{k+1} - y_{k+1}) \\ & + 2t_{k+1}(x_{k+1} - y_{k+1})^T \sigma_{k+1} H_{k+1}(t_{k+1} y_{k+1} - (t_{k+1} - 1)x_k - x_*). \end{aligned}$$

By applying (4.7) with

$$a := \sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}} t_{k+1} y_{k+1}, \quad b := \sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}} t_{k+1} x_{k+1}, \quad c := \sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}}((t_{k+1} - 1)x_k + x_*),$$

the last inequality can be written as

$$\begin{aligned} 2(\sigma_k t_k^2 v_k - \sigma_{k+1} t_{k+1}^2 v_{k+1}) \geq\ & \|\sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}} t_{k+1} x_{k+1} - \sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}}((t_{k+1} - 1)x_k + x_*)\|^2 \\ & - \|\sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}} t_{k+1} y_{k+1} - \sqrt{\sigma_{k+1}} H_{k+1}^{\frac{1}{2}}((t_{k+1} - 1)x_k + x_*)\|^2. \end{aligned}$$

Hence, by using the definition of y_{k+1} and u_k, we have

$$2(\sigma_k t_k^2 v_k - \sigma_{k+1} t_{k+1}^2 v_{k+1}) \geq u_{k+1}^T \sigma_{k+1} H_{k+1} u_{k+1} - u_k^T \sigma_{k+1} H_{k+1} u_k.$$

Now, based on (4.8a), we have

$$u_k^T \sigma_k H_k u_k \geq u_k^T \sigma_{k+1} H_{k+1} u_k,$$

which implies

$$2(\sigma_k t_k^2 v_k - \sigma_{k+1} t_{k+1}^2 v_{k+1}) \geq u_{k+1}^T \sigma_{k+1} H_{k+1} u_{k+1} - u_k^T \sigma_k H_k u_k.$$

□

Now, we are ready to state and prove the convergence rate result.

Theorem 4.7 *The sequence of iterates x_k, generated by Algorithm 5, satisfies*

$$F(x_k) - F(x_*) \leq \frac{\|x_0 - x_*\|^2}{2\sigma_k t_k^2}.$$

Proof By setting $t_1 = 1$, using the definition of u_k at $k = 1$, which is $u_1 = x_1 - x_*$, and also considering the positive definiteness of H_k for all $k \geq 1$, it follows from Lemma 4.6 that

$$2\sigma_k t_k^2 v_k \leq 2\sigma_k t_k^2 v_k + \sigma_k u_k^T H_k u_k \leq 2\sigma_1 t_1^2 v_1 + (x_1 - x_*)^T \sigma_1 H_1 (x_1 - x_*). \tag{4.9}$$

Setting $x = x_*$, $v = y_1 = x_0$, $p_H(v) = x_1$, $t_1 = 1$, and $H = H_1$ in (4.6) implies

$$-2v_1 \geq (x_1 - x_*)^T H_1 (x_1 - x_*) - (x_0 - x_*)^T H_1 (x_0 - x_*).$$

Multiplying the above by σ_1 gives

$$2\sigma_1 v_1 + (x_1 - x_*)^T \sigma_1 H_1 (x_1 - x_*) \leq (x_0 - x_*)^T \sigma_1 H_1 (x_0 - x_*).$$

By using inequality (4.9), we have

$$2\sigma_k t_k^2 v_k \leq (x_0 - x_*)^T \sigma_1 H_1 (x_0 - x_*).$$

Finally, by setting $\sigma_1 = 1$ and $H_1 = I$, we obtain

$$v_k \leq \frac{\|x_0 - x_*\|^2}{2\sigma_k t_k^2},$$

which completes the proof. □

Now, based on the result of Theorem 4.7, in order to obtain the rate of convergence of $\mathcal{O}(1/k^2)$ for Algorithm 5, it is sufficient to show that

$$\sigma_k t_k^2 \geq \psi k^2,$$

for some constant $\psi > 0$. The next result is a simple consequence of the relation (4.8b), or equivalently (4.1a).

Lemma 4.8 *The sequence $\{\sigma_k\}$ generated by Algorithm 5 satisfies*

$$\sigma_k t_k^2 \geq \left(\frac{\sum_{i=1}^k \sqrt{\sigma_i}}{2} \right)^2.$$

Proof We can prove this lemma by using induction. Trivially, for $k = 1$, since $t_1 = 1$, the inequality holds. As the induction assumption, assume that for $k > 1$, we have $\sigma_k t_k^2 \geq \left(\frac{\sum_{i=1}^k \sqrt{\sigma_i}}{2} \right)^2$. Since (4.1a) holds for all k, it follows that

$$t_{k+1} = \frac{1}{2} + \sqrt{\frac{1}{4} + (\frac{\sigma_k}{\sigma_{k+1}}) t_k^2} \geq \frac{1}{2} + \sqrt{\frac{\sigma_k}{\sigma_{k+1}}} t_k.$$

Multiplying by $\sqrt{\sigma_{k+1}}$ implies

$$\sqrt{\sigma_{k+1}} t_{k+1} \geq \frac{\sqrt{\sigma_{k+1}}}{2} + \sqrt{\sigma_k} t_k.$$

Finally, by using induction assumption, we will have have

$$\sqrt{\sigma_{k+1}} t_{k+1} \geq \frac{\sqrt{\sigma_{k+1}}}{2} + \frac{\sum_{i=1}^k \sqrt{\sigma_i}}{2} = \frac{\sum_{i=1}^{k+1} \sqrt{\sigma_i}}{2}.$$

□

Hence, if we assume that the sequence $\{\sigma_k\}$ is bounded below by a positive constant $\underline{\sigma}$, i.e., $\sigma_k \geq \underline{\sigma}$, we can establish the desired bound on $\sigma_k t_k^2$, as stated in the following theorem.

Theorem 4.9 *If for all iterations of Algorithm 5 we have $\sigma_k \geq \underline{\sigma}$, then for all k*

$$F(x_k) - F(x_*) \leq \frac{2\|x_0 - x_*\|^2}{\underline{\sigma} k^2}. \tag{4.10}$$

Proof Under the assumption $\sigma_k \geq \underline{\sigma}$, we will have

$$\left(\frac{\sum_{i=1}^k \sqrt{\sigma_i}}{2}\right)^2 \geq \frac{k^2 \underline{\sigma}}{4},$$

and consequently, by using Lemma 4.8, we obtain

$$\sigma_k t_k^2 \geq \frac{k^2 \underline{\sigma}}{4}.$$

Then, by using Theorem 4.7, we have the desired rate of convergence of $\mathcal{O}(1/k^2)$ as stated in (4.10). □

The assumption of the existence of a bounded sequence $\{\sigma_k\}$ such that $\sigma_k \geq \underline{\sigma}$ and (4.8b) holds may not be satisfied when we use a general approximate Hessian. To illustrate this, consider the following simple sequence of matrices:

$$H_{2k} = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, \quad H_{2k+1} = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}.$$

Clearly, $\sigma_{2k+1} \leq \sigma_{2k}/10$ and $\sigma_{2k} \leq \sigma_{2k-1}/10$, and hence $\sigma_k \leq 10^{-k}$. In this case, based on the result of Theorem 4.7, we cannot guarantee any convergence result. Some convergence result can still be attained, when $\sigma_k \to 0$, for example, if $\sigma_k \geq \underline{\sigma}/k$, as we show in the following relaxed version of Theorem 4.9.

Theorem 4.10 *If for all iterations of Algorithm 5 we have $\sigma_k \geq \underline{\sigma}/k$, then for all k*

$$F(x_k) - F(x_*) \leq \frac{2\|x_0 - x_*\|^2}{\underline{\sigma} k}. \tag{4.11}$$

Proof From $\sigma_k \geq \underline{\sigma}/k$, we will have

$$\left(\frac{\sum_{i=1}^k \sqrt{\sigma_i}}{2}\right)^2 \geq \frac{k \underline{\sigma}}{4},$$

and consequently, by using Lemma 4.8, we obtain

$$\sigma_k t_k^2 \geq \frac{k \underline{\sigma}}{4}.$$

Then, by using Theorem 4.7, we have (4.11). □

The above theorem shows that if σ_k converges to zero, but not faster than $1/k$, then our *APQNA* method may loose its accelerated rate of convergence, but still converges at least at the same rate as *PQNA*. Establishing lower bounds of σ_k for different choices of Hessian estimates is a nontrivial task and is the subject of future research. As we will demonstrate in our computational section, *APQNA* with L-BFGS Hessian approximation does not seem to have any practical advantage over its nonaccelerated counterpart, however it is clearly convergent.

We can establish the accelerated rate of Algorithm 6, since in this case we can guarantee a lower bound on σ_k, due to the restricted nature of the H_k matrices.

Lemma 4.11 *In Algorithm 6, let $mI \preceq H$, then $\sigma_k \geq \beta m/L$ and hence the convergence rate of $\mathcal{O}(1/k^2)$ is achieved.*

Proof In Algorithm 6, we define $H_k = \frac{1}{\sigma_k} H$. The sufficient decrease condition $F(p_{H_k}(y_k)) \leq Q_{H_k}(p_{H_k}(y_k), y_k)$, is satisfied for any $H_k \succeq LI$, hence it is satisfied for any $H_k = \frac{1}{\sigma_k} H$ with $\sigma_k \leq m/L$. By the mechanism of Step 3 in Algorithm 6, we observe that for all k, we have $\sigma_k \geq \beta m/L$. Let us note now that Algorithm 6 is a special case of Algorithm 5, hence all the above results, in particular Theorem 4.7 and Lemma 4.8 hold. Consequently, based on Theorem 4.9, the desired convergence rate of $\mathcal{O}(1/k^2)$ for Algorithm 6 is obtained. □

Remark 4.12 We have studied only the exact variant of *APQNA* in this section. Incorporating inexact subproblem solutions, as was done for *APQNA* in the previous section, is relatively straightforward following the techniques for inexact *APGA*, [20]. It is easy to show that if the exact algorithm has the accelerated convergence rate, then the inexact counterpart, with subproblems solved by a linearly convergent method, such as randomized coordinate descent, inherits this convergence rate. However, using the relaxed sufficient decrease condition does not apply here as it does not preserve the accelerated convergence rate.

In the next section, we present the numerical results comparing the performance of Algorithm 5 and Algorithm 6 to their nonaccelerated counterparts, to see how much practical acceleration is achieved.

5 Numerical Experiments

In this section, we investigate the practical performance of several algorithms discussed in this work, applied to the sparse logistic regression problem

$$\min_w \{F(w) := \frac{1}{m}\sum_{i=1}^{m} \log(1+\exp(-y_i \cdot w^T x_i)) + \lambda\|w\|_1,\ w \in \mathbb{R}^n\},$$

where $f(w) = \sum_{i=1}^m \log(1+\exp(-y_i \cdot w^T x_i))$ is the average logistic loss function and $g(w) = \lambda\|w\|_1$ is the ℓ_1-regularization function. The input data for this problem is a set of m training data points, $x_i \in \mathbb{R}^n$, and corresponding labels $y_i \in \{-1, +1\}$, for $i = 1, 2, \ldots, m$.

The algorithms that we compare here are as follows:

- Accelerated Proximal Gradient Algorithm (*APGA*), proposed in [1], (also known as FISTA),
- Proximal Quasi-Newton Algorithm with Fixed Hessian approximation (*PQNA-FH*),
- Accelerated Proximal Quasi-Newton Algorithm with Fixed Hessian approximation (*APQNA-FH*),
- Proximal Quasi-Newton Algorithm with L-BFGS Hessian approximation (*PQNA-LBFGS*), proposed in [19], and
- Accelerated Proximal Quasi-Newton Algorithm with L-BFGS Hessian approximation (*APQNA-LBFGS*).

In *PQNA-FH* and *APQNA-FH*, we set $H_k = \frac{1}{\sigma_k}H$, where H is a positive definite matrix computed via applying L-BFGS updates over the first few iterations of the algorithm which then is fixed for all remaining iterations. On the other hand, *PQNA-LBFGS* and *APQNA-LBFGS* employ the L-BFGS updates to compute Hessian estimates throughout the algorithm. In all of the above algorithms, we use the coordinate descent scheme, as described in [19], to solve the subproblems inexactly. According to the theory in [19], *PQNA-FH* and *PQNA-LBFGS* converge at the rate of $\mathcal{O}(1/k)$. If f is strongly convex (which depends on the problem data), then according to Theorem 3.2, *PQNA-FH* and *PQNA-LBFGS* converge at a linear rate. By Lemma 4.11, in *APQNA-FH*, condition $\sigma_k H_k \preceq \sigma_{k-1}H_{k-1}$ holds automatically and the algorithm converges at the rate of $\mathcal{O}(1/k^2)$. On the other hand, for *APQNA-LBFGS*, condition $\sigma_k H_k \preceq \sigma_{k-1}H_{k-1}$ has to be enforced. We have tested various implementations that ensure this condition and none have produced a practical approach. We then chose to set $\theta_k = 1$ and relax the condition $\sigma_k H_k \preceq \sigma_{k-1}H_{k-1}$. The resulting algorithm is practical and is empirically convergent, but as we will see does not provide an improvement over *PQNA-LBFGS*.

Throughout all of our experiments, we initialize the algorithms with $w_0 = \mathbf{0}$ and we set the regularization parameter $\lambda = 0.001$. Each algorithm terminates whenever $\|\partial F\|_\infty \leq 10^{-5}\|\partial F_0\|_\infty$. In terms of the stopping criteria of subproblems solver at i-th iteration, we performed the coordinate descent method for $r(i)$ steps, so that $r(i) > \min(1000, i/3)$, as long as the generated step is longer than 10^{-16}. In *APQNA-FH* and *PQNA-FH*, in order to construct the fixed matrix H, we apply the L-BFGS scheme by using the information from the first $\bar{k}$ (with $\bar{k}$ chosen between 1 and 10) iterations and then use that fixed matrix through the rest of the algorithm. Finally, to construct the sequence $\{\sigma_k\}$, we set $\sigma_0 = 1$ and $\sigma_{k+1}^0 = 1.015\sigma_k$. The information on the data sets used in our tests is summarized in Table 5.1. These data sets are available through UCI machine learning repository [2].

The algorithms are implemented in MATLAB R2014b and computations were performed on the COR@L computational cluster of the ISE department at Lehigh, consisting of 16-cores AMD Operation, 2.0 GHz nodes with 32 Gb of memory.

First, in order to demonstrate the effect of using even limited Hessian information within an accelerated method, we compared the performance of *APQNA-FH* and *APGA*, both in terms of the number of iterations and the total solution time, see the results in Table 5.2.

[2] http://archive.ics.uci.edu/ml/

Table 5.1: data information

Instance	d	N	Description
a9a	123	32561	census income dataset
mnist	782	100000	handwritten digit recognition
connect-4	126	10000	win versus loss recognition
HAPT	561	7767	human activities and postural transitions recognition

Table 5.2: *APQNA-FH* vs. *APGA* in terms of number of iterations and solution time

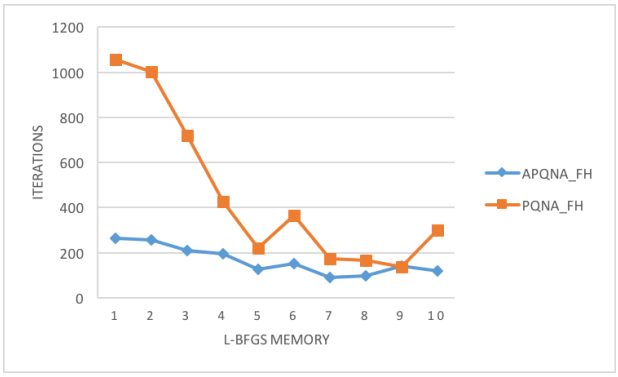
a9a							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APGA	40	3.4891e-01	80	3.4730e-01	862	1.9536e+01	3.4703e-01
APQNA-FH	40	3.4706e-01	80	3.4703e-01	121	5.5294e+00	3.4703e-01
mnist							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APGA	48	9.1506e-02	96	9.0206e-02	1202	5.1337e+02	8.9695e-02
APQNA-FH	48	8.9754e-02	96	8.9699e-02	144	9.9121e+01	8.9695e-02
connect-4							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APGA	92	3.8284e-01	184	3.7777e-01	3045	4.6563e+01	3.7682e-01
APQNA-FH	92	3.7701e-01	184	3.7683e-01	278	2.0528e+01	3.7682e-01
HAPT							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APGA	222	8.5415e-02	444	7.7179e-02	13293	1.3860e+03	7.1511e-02
APQNA-FH	222	7.2208e-02	444	7.1524e-02	677	1.5315e+02	7.1511e-02

Based on the results shown in Table 5.2, we conclude that *APQNA-FH* consistently dominates the *APGA*, both in terms of the number of function evaluations and also in terms of the total solution time.

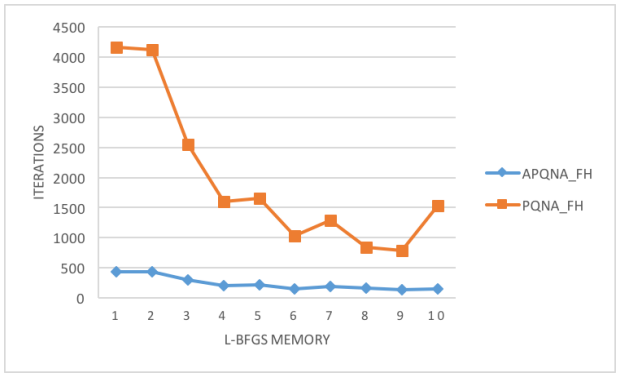
It is worth mentioning that although in terms of computational effort, each iteration of *APGA* is cheaper than each iteration of *APQNA-FH*, the total solution time of *APQNA-FH* is significantly less than *APGA*, due to the smaller number of iterations of *APQNA-FH* compared to *APGA*.

The next experiment is to compare the performance of *APQNA-FH* and *PQNA-FH* to observe the effect of acceleration in the fixed matrix setting. This comparison is done in terms of the number of iterations and the number of function evaluations, and is shown in Figure 5.1 and Figure 5.2, respectively. The subproblem solution time is the same for both algorithms. As we can see in Figure 5.1, in terms of the number of iterations, *APQNA-FH* dominates *PQNA-FH*, for a range of memory sizes of L-BFGS which have been used to compute matrix H. Moreover, as is seen in Figure 5.2, *APQNA-FH* dominates *PQNA-FH*, in terms of the number of function evaluations, even though each iteration of *APQNA-FH* requires two function evaluations, because of the nature of the accelerated scheme. This shows that *APQNA-FH* achieves practical acceleration compared to *PQNA-FH*, as supported by the theory in the previous section.

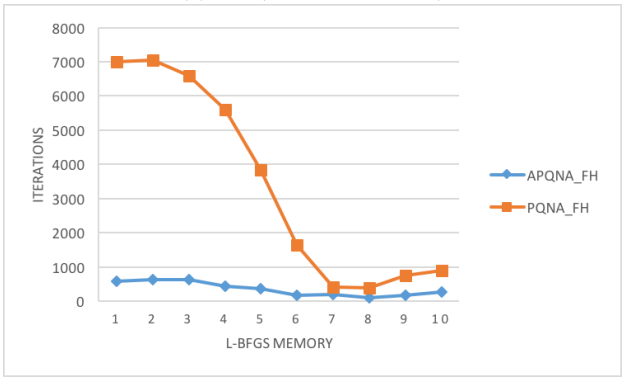
Next, we compare the performance of *APQNA-FH* versus *APQNA-LBFGS* to compare the effect of using the fixed approximate Hessian $H_k = \frac{1}{\sigma_k} H$, which satisfies condition $\sigma_k H_k \preceq \sigma_{k-1} H_{k-1}$ versus using variable Hessian estimates computed via L-BFGS method at each iteration, while relaxing condition $\sigma_k H_k \preceq \sigma_{k-1} H_{k-1}$. Table 5.3 shows the results of this comparison, obtained based on the best choices of memory size for L-BFGS, in particular $\bar{k} = 8$ and $\bar{k} = 9$, respectively. As we can see, these two algorithms are competitive both in terms of the number of iterations and also the total solution time. Since *APQNA-FH* does not use the local information of function f to approximate H_k, it often takes more iterations than *APQNA-LBFGS*, which constantly updates H_k matrices. On the other hand, since *APQNA-FH* does not require additional computational effort to evaluate H_k, hence one iteration of this algorithm is cheaper than one iteration of *APQNA-LBFGS*, which causes the competitive total solution time.



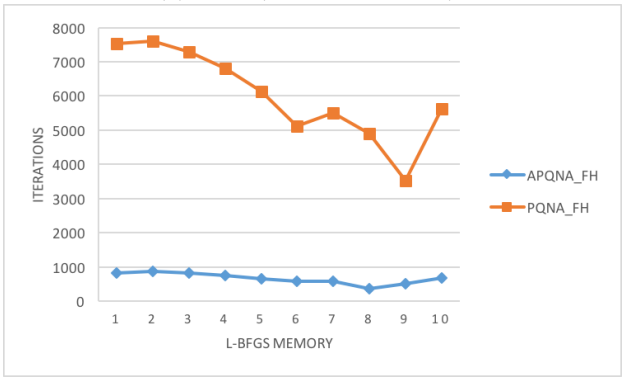
(a) a9a (d=123, m=32561)



(b) mnist (d=782, m=100000)



(c) connect-4 (d=126, m=10000)



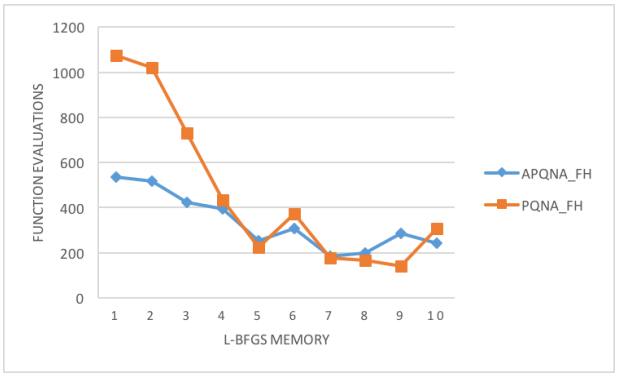
(d) HAPT (d=561, m=7767)

Fig. 5.1: *APQNA-FH* vs. *PQNA-FH* in terms of number of iterations

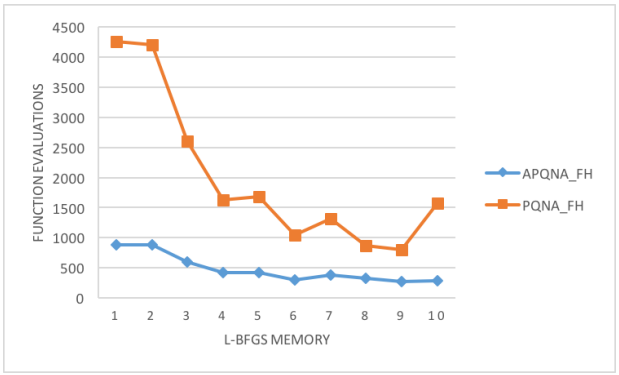
Table 5.3: *APQNA-FH* vs. *PQNA-LBFGS* in terms of number of iterations and solution time

a9a							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APQNA-LBFGS	20	3.4760e-01	40	3.4703e-01	64	2.8328e+00	3.4703e-01
APQNA-FH	20	3.4763e-01	40	3.4704e-01	99	4.3320e+00	3.4703e-01
mnist							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APQNA-LBFGS	50	8.9713e-02	100	8.9695e-02	148	1.0425e+02	8.9695e-02
APQNA-FH	50	8.9797e-02	100	8.9698e-02	160	1.1514e+02	8.9695e-02
connect-4							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APQNA-LBFGS	30	3.7769e-01	60	3.7688e-01	144	8.3596e+00	3.7682e-01
APQNA-FH	30	3.7689e-01	60	3.7682e-01	93	3.9594e+00	3.7682e-01
HAPT							
Algorithm	iter	Fval	iter	Fval	final iter	final time(sec.)	final Fval.
APQNA-LBFGS	120	7.1860e-02	240	7.1519e-02	356	1.0474e+02	7.1511e-02
APQNA-FH	120	7.2134e-02	240	7.1523e-02	376	6.8952e+01	7.1511e-02

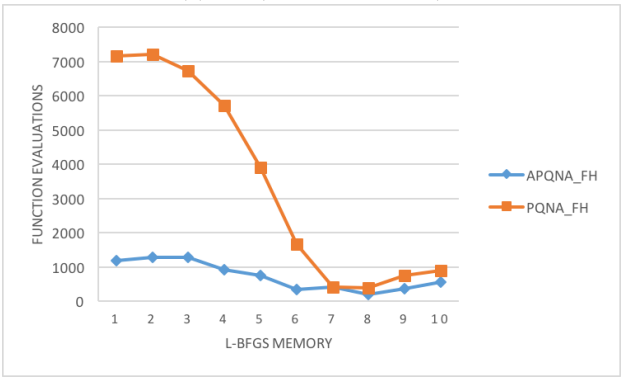
Finally, we compare *APQNA-LBFGS* and *PQNA-LBFGS*, to demonstrate the effect of using an accelerated scheme in the quasi-Newton type proximal algorithms. The results of this comparison are shown in Figure 5.3 and Figure 5.4 in terms of the number of iterations and the number of function evaluations, respectively, for different memory sizes of L-BFGS Hessian approximation.



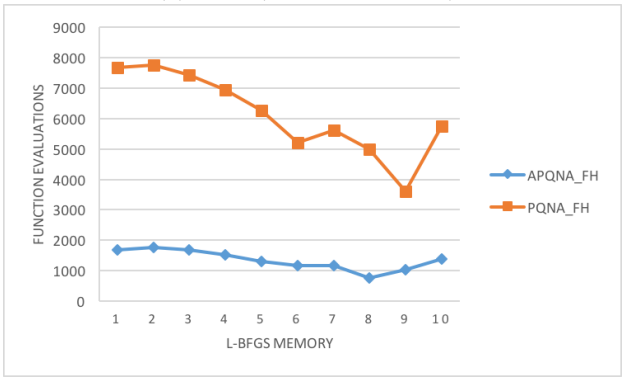
(a) a9a (d=123, m=32561)



(b) mnist (d=782, m=100000)

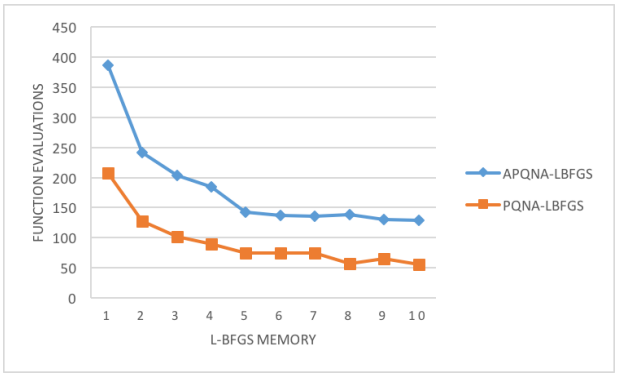


(c) connect-4 (d=126, m=10000)

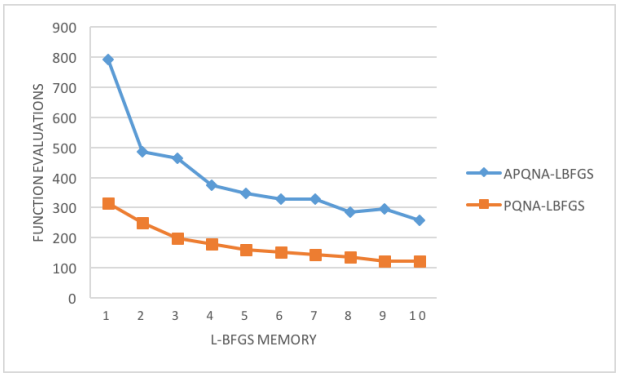


(d) HAPT (d=561, m=7767)

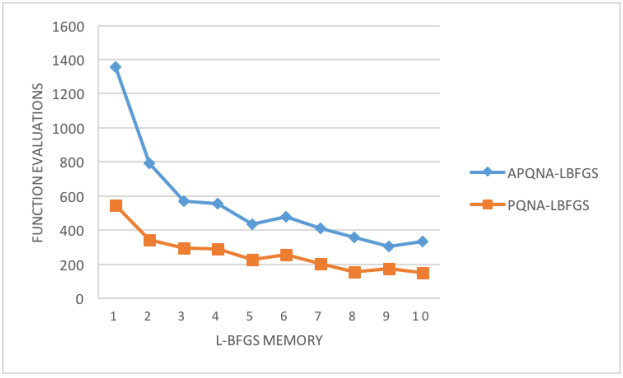
Fig. 5.2: *APQNA-FH* vs. *PQNA-FH* in terms of number of function evaluations



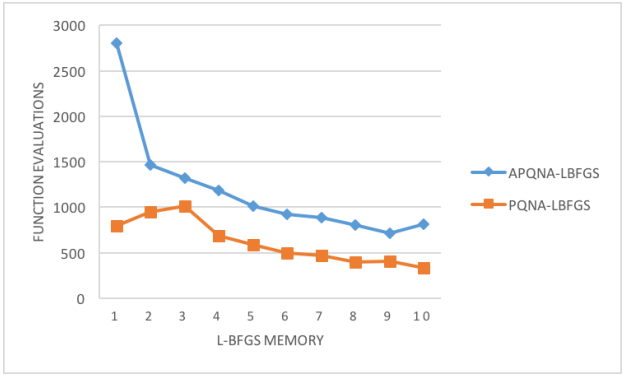
(a) a9a (d=123, m=32561)



(b) mnist (d=782, m=100000)

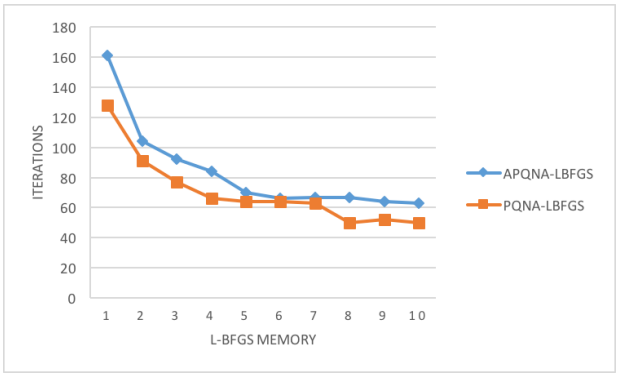


(c) connect-4 (d=126, m=10000)

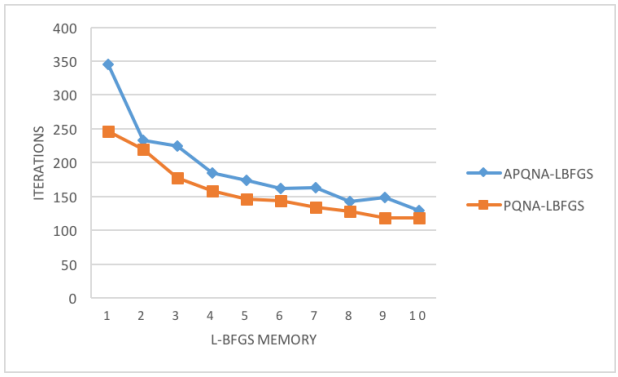


(d) HAPT (d=561, m=7767)

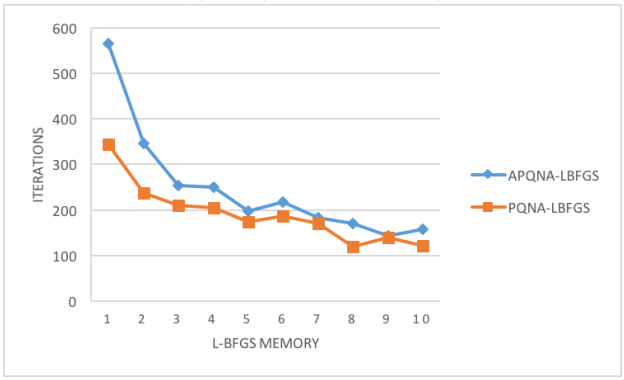
Fig. 5.4: *APQNA-LBFGS* vs. *PQNA-LBFGS* in terms of number of function evaluations



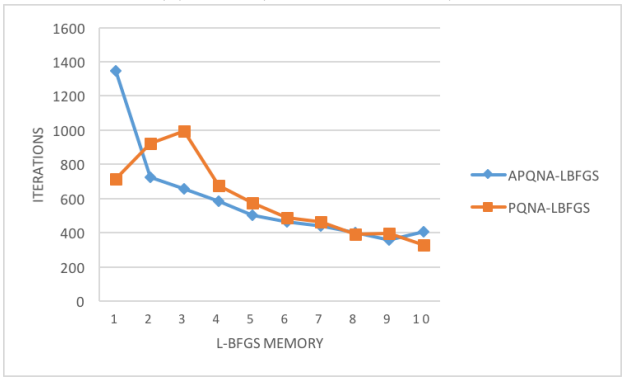
(a) a9a (d=123, m=32561)



(b) mnist (d=782, m=100000)



(c) connect-4 (d=126, m=10000)



(d) HAPT (d=561, m=7767)

Fig. 5.3: *APQNA-LBFGS* vs. *PQNA-LBFGS* in terms of number of iterations

Clearly, as we can see in Figure 5.3 and 5.4, not only the accelerated scheme does not achieve practical acceleration compared to *PQNA-LBFGS* in terms of the number of iterations, but it is also inferior in terms of the number of function evaluations, since every iteration requires two function evaluations. Thus, we believe that the practical experiments support our theoretical analysis in that applying acceleration scheme in the case of variable Hessian estimates may not result in a faster algorithm.

6 Conclusion

In this paper, we established a linear convergence rate of *PQNA* proposed in [19] under strong convexity. To our knowledge, this is the first such a result, for proximal quasi-Newton type methods, which have lately been popular in the literature. We also show that this convergence rate is preserved when subproblems are solved inexactly. We provide a simple and practical rule for the number of inner iterations that guarantee sufficient accuracy of subproblem solutions. Moreover, we allow a relaxed sufficient decrease condition during backtracking, which preserves the convergence rate, while it is known to improve the practical performance of the algorithm.

Furthermore, we presented a variant of *APQNA* as an extension of *PQNA*. We have shown that this algorithm has the convergence rate of $\mathcal{O}(1/k^2)$ under a strong condition on the Hessian estimates, which can not always be guaranteed in practice. We have shown that this condition holds when Hessian estimates are a multiple of a fixed matrix, which is computationally less expensive than the more common methods, such as the L-BFGS scheme. Although, this proposed algorithm has the same rate of convergence as the classic *APGA*, it is significantly faster in terms of the final number of iterations and also the total solution time. Based on the theory, using L-BFGS Hessian approximation, may result in worse convergence rate, however, our computational results show that the practical performance is about the same as that while the fixed matrix. On the other hand, although in these two algorithms, we are applying the accelerated scheme, their practical performances are inferior to that of *PQNA-LBFGS*, which does not use any accelerated scheme and potentially has a slower sublinear rate of

convergence in the absence of strong convexity. We conclude that using variable Hessian estimates is the most efficient approach, and will result in the linear convergence rate in the presence of strong convexity, but that a standard accelerated scheme is not useful in this setting. Exploring other possibly more effective accelerated schemes for the proximal quasi-Newton methods is the subject of future research.

References

1. A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM*, 2:183–202, 2009.
2. R. Byrd, J. Nocedal, and F. Oztoprak. An inexact successive quadratic approximation method for convex ℓ_1-regularized optimization. *Technical Report*, 2013.
3. R. H Byrd, J. Nocedal, and R. B Schnabel. Representations of quasi Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.
4. D. Drusvyatskiy and A. S. Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Technical Report*, 2011.
5. H. Ghanbari and K. Scheinberg. Optimization algorithms in machine learning. *Doctoral Dissertation*, 2016.
6. C. J. Hsieh, M. Sustik, I. Dhilon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. *NIPS*, pages 2330–2338, 2011.
7. K. Jiang, D. Sun, and K. Toh. An inexact accelerated proximal gradient method for large scale linearly constrained convex SDP. *SIAM*, 22:1042–1064, 2012.
8. J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for convex optimization. *Technical Report*, 2012.
9. A. Nemirovski and D. Yudin. Informational complexity and efficient methods for solution of convex extremal problems. *J. Wiley and Sons, New York*, 1983.
10. Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.
11. Y. E. Nesterov. *Introductory Lectures on Convex Programming: A Basic Course*. Springer, 2004.
12. Y. E. Nesterov. Smooth minimization for non-smooth functions. *Mathematical Programming*, 103:127–152, 2005.
13. Y. E. Nesterov. Gradient methods for minimizing composite objective function. *Mathematical Programming*, 140:125–161, 2013.
14. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, NY, USA, 2nd edition, 2006.
15. P. A. Olsen, F. Oztoprak, J. Nocedal, and S. J. Rennie. Newton-like methods for sparse inverse covariance estimation. *NIPS*, pages 764–772, 2012.
16. P. Richtarik and M. Takac. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144:1–38, 2014.
17. K. Scheinberg, D. Goldfarb, and X. Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundation of Mathematics*, 14:389–417, 2014.
18. K. Scheinberg and I. Rish. A greedy coordinate ascent method for sparse inverse covariance selection problem. *SINCO, Technical Report*, 2009.
19. K. Scheinberg and X. Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Mathematical Programming*, 2016.
20. M. Schmidt, N. L. Roux, and F. Bach. Convergence rate of inexact proximal-gradient method for convex optimization. *NIPS*, pages 1458–1466, 2011.
21. M. Schmidt, N. L. Roux, and F. Bach. Supplementary material for the paper convergence rates of inexact proximal-gradient methods for convex optimization. *NIPS*, 2011.
22. S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *ICML*, pages 929–936, 2009.
23. S. Sra, S. Nowozin, and S.J. Wright. *Optimization for Machine Learning*. Mit Pr, 2011.
24. R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B, Methodological*, 58:267–288, 1996.
25. P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *Technical Report*, 2008.
26. S. Villa, S. Salzo, L. Baldassarre, and A. Verri. Accelerated and inexact forward-backward algorithms. *SIAM*, 23:1607–1633, 2011.
27. G. X. Yuan, K. W. Chang, C. J. Hsieh, and C. J. Lin. A comparison of optimization methods and software for large-scale ℓ_1-regularized linear classification. *JMLR*, 11:3183–3234, 2010.