Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present the first accelerated randomized algorithm for solving linear systems in Euclidean spaces. One essential problem of this type is the matrix inversion problem. In particular, our algorithm can be specialized to invert positive definite matrices in such a way that all iterates (approximate solutions) generated by the algorithm are positive definite matrices themselves. This opens the way for many applications in the field of optimization and machine learning. As an application of our general theory, we develop the *first accelerated (deterministic and stochastic) quasi-Newton updates*. Our updates lead to provably more aggressive approximations of the inverse Hessian, and lead to speed-ups over classical non-accelerated rules in numerical experiments. Experiments with empirical risk minimization show that our rules can accelerate training of machine learning models.

1 Introduction

Consider the optimization problem

$$\min_{w\in\mathbb{R}^n} f(w), \tag{1}$$

and assume f is sufficiently smooth. A new wave of second order stochastic methods are being developed with the aim of solving large scale optimization problems. In particular, many of these new methods are based on stochastic BFGS updates [26, 32, 17, 18, 33, 4]. Here we develop a new stochastic accelerated BFGS update that can form the basis of new stochastic quasi-Newton methods.

Another approach to scaling up second order methods is to use randomized *sketching* to reduce the dimension, and hence the complexity of the Hessian and the updates involving the Hessian [23, 35], or *subsampled* Hessian matrices when the objective function is a sum of many loss functions [2, 1].

The starting point for developing second order methods is arguably Newton's method, which performs the iterative process

$$w_{k+1} = w_k - (\nabla^2 f(w_k))^{-1}\nabla f(w_k), \tag{2}$$

where $\nabla^2 f(w_k)$ and $\nabla f(w_k)$ are the Hessian and gradient of f, respectively. However, it is inefficient for solving large scale problems as it requires the computation of the Hessian and then solving a linear system in each iteration. Several methods have been developed to address this issue, based on the idea of approximating the exact update.

Quasi-Newton methods, in particular the BFGS [3, 7, 8, 27], have been the leading optimization algorithm in various fields since the late 60's until the rise of big data, which brought a need for simpler first order algorithms. It is well known that Nesterov's acceleration [20] is a reliable way

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

to speed up first order methods. However until now, acceleration techniques have been applied exclusively to speeding up gradient updates. In this paper we present an accelerated BFGS algorithm, opening up new applications for acceleration. The acceleration in fact comes from an accelerated algorithm for inverting the Hessian matrix.

To be more specific, recall that quasi-Newton rules aim to maintain an estimate of the inverse Hessian X_k, adjusting it every iteration so that the inverse Hessian acts appropriately in a particular direction, while enforcing symmetry:

$$X_k(\nabla f(w_k) - \nabla f(w_{k-1})) = w_k - w_{k-1}, \qquad X_k = X_k^\top. \tag{3}$$

A notable research direction is the development of stochastic quasi-Newton methods [11], where the estimated inverse is equal to the true inverse over a subspace:

$$X_k \nabla^2 f(w_k) S_k = S_k, \qquad X_k = X_k^\top, \tag{4}$$

where $S_k \in \mathbb{R}^{n\times\tau}$ is a randomly generated matrix.

In fact, (4) can be seen as the so called sketch-and-project iteration for inverting $\nabla^2 f(w_k)$. In this paper we first develop the accelerated algorithm for inverting positive definite matrices. As a direct application, our algorithm can be used as a primitive in quasi-Newton methods which results in a novel accelerated (stochastic) quasi-Newton method of the type (4). In addition, our acceleration technique can also be incorporated in the classical (non stochastic) BFGS method. This results in the accelerated BFGS method. Whereas the matrix inversion contribution is accompanied by strong theoretical justifications, this does not apply to the latter. Rather, we verify the effectiveness of this new accelerated BFGS method through numerical experiments.

1.1 Sketch-and-project for linear systems

Our accelerated algorithm can be applied to more general tasks than only inverting matrices. In its most general form, it can be seen as an accelerated version of a *sketch-and-project* method in Euclidean spaces which we present now. Consider a linear system $Ax = b$ such that $b \in \mathbf{Range}\,(A)$. One step of the sketch-and-project algorithm reads as:

$$x_{k+1} = \text{argmin}_x \ \|x_k - x\|_B^2 \quad \text{subject to} \quad S_k^\top A x = S_k^\top b, \tag{5}$$

where $\|x\|_B^2 = \langle Bx, x\rangle$ for some $B \succ 0$ and S_k is a random sketching matrix sampled i.i.d at each iteration from a fixed distribution.

Randomized Kaczmarz [13, 30] was the first algorithm of this type. In [12], this sketch-and-project algorithm was analyzed in its full generality. Note that the dual problem of (5) takes the form of a quadratic minimization problem [10], and randomized methods such as coordinate descent [19, 34], random pursuit [29, 28] or stochastic dual ascent [10] can thus also be captured as special instances of this method. Richtárik and Takáč [25] adopt a new point of view through a theory of stochastic reformulations of linear systems. In addition, they consider the addition of a relaxation parameter, as well as mini-batch and accelerated variants. Acceleration was only achieved for the expected iterates, and not in the L2 sense as we do here. We refer to Richtárik and Takáč [25] for interpretation of sketch-and-project as stochastic gradient descent, stochastic Newton, stochastic proximal point method, and stochastic fixed point method.

Gower [11] observed that the procedure (5) can also be applied to find the inverse of a matrix. Assume the optimization variable itself is a matrix, $x = X$, $b = I$, the identity matrix, then sketch-and-project converges (under mild assumptions) to a solution of $AX = I$. Even the symmetry constraint $X = X^\top$ can be incorporated into the sketch-and-project framework since it is a linear constraint.

There has been recent development in speeding up the sketch-and-project method using the idea of Nesterov's acceleration [20]. In [15] an accelerated Kaczmarz algorithm was presented for special sketches of rank one. Arbitrary sketches of rank one where considered in [29], block sketches in [21] and recently, Tu and coathors [31] developed acceleration for special sketching matrices, assuming the matrix A is square. This assumption, along with any assumptions on A, was later dropped in [24]. Another notable way to accelerate the sketch-and-project algorithm is by using momentum or stochastic momentum [16].

We build on recent work of Richtárik and Takáč [24] and further extend their analysis by studying accelerated sketch-and-project in general Euclidean spaces. This allows us to deduce the result for

matrix inversion as a special case. However, there is one additional caveat that has to be considered for the intended application in quasi-Newton methods: ideally, all iterates of the algorithm should be symmetric positive definite matrices. This is not the case in general, but we address this problem by constructing special sketch operators that preserve symmetry and positive definiteness.

2 Contributions

We now present our main contributions.

Accelerated Sketch and Project in Euclidean Spaces. We generalize the analysis of an accelerated version of the sketch-and-project algorithm [24] to linear operator systems in Euclidean spaces. We provide a self-contained convergence analysis, recovering the original results in a more general setting.

Faster Algorithms for Matrix Inversion. We develop an accelerated algorithm for inverting positive definite matrices. This algorithm can be seen as a special case of the accelerated sketch-and-project in Euclidean space, thus its convergence follows from the main theorem. However, we also provide a different formulation of the proof that is specialized to this setting. Similarly to [31], the performance of the algorithm depends on two parameters μ and ν that capture spectral properties of the input matrix and the sketches that are used. Whilst for the non-accelerated sketch-and-project algorithm for matrix inversion [11] the knowledge of these parameters is not necessary, they need to be given as input to the accelerated scheme. When employed with the correct choice of parameters, the accelerated algorithm is always faster than the non-accelerated one. We also provide a theoretical rate for sub-optimal parameters μ, ν, and we perform numerical experiments to argue the choice of μ, ν in practice.

Randomized Accelerated Quasi-Newton. The proposed iterative algorithm for matrix inversion is designed in such a way that each iterate is a symmetric matrix. This means, we can use the generated approximate solutions as estimators for the inverse Hessian in quasi-Newton methods, which is a direct extension of stochastic quasi-Newton methods. To the best of our knowledge, this yields the first accelerated (stochastic) quasi-Newton method.

Accelerated Quasi-Newton. In the standard BFGS method the updates to the Hessian estimate are not chosen randomly, but deterministically. Based on the intuition gained from the accelerated random method, we propose an accelerated scheme for BFGS. The main idea is that we replace the random sketching of the Hessian with a deterministic update. The theoretical convergence rates do not transfer to this scheme, but we demonstrate by numerical experiments that it is possible to choose a parameter combination which yields a slightly faster convergence. We believe that the novel idea of accelerating BFGS update is extremely valuable, as until now, acceleration techniques were only considered to improve gradient updates.

2.1 Outline

Our accelerated sketch-and-project algorithm for solving linear systems in Euclidean spaces is developed and analyzed in Section 3, and is used later in Section 4 to analyze an accelerated sketch-and-project algorithm for matrix inversion. The accelerated sketch-and-project algorithm for matrix inversion is then used to accelerate the BFGS update, which in term leads to the development of an accelerated BFGS optimization method. Lastly in Section 5, we perform numerical experiments to gain different insights into the newly developed methods. Proofs of all results and additional insights can be found in the appendix.

3 Accelerated Stochastic Algorithm for Matrix Inversion

In this section we propose an accelerated randomized algorithm to solve linear systems in Euclidean spaces. This is a very general problem class and it comprises for instance also the matrix inversion problem. Thus, we will use the result of this section later to analyze our newly proposed matrix inversion algorithm, which we then use to estimate the inverse of the Hessian within a quasi-Newton method.[1]

[1] Quasi-Newton methods do not compute an exact matrix inverse, rather, they only compute an incremental update. Thus, it suffices to apply *one step* of our proposed scheme per iteration. This will be detailed in Section 4.

Let $\mathcal{X}$ and $\mathcal{Y}$ be finite dimensional Euclidean spaces and let $\mathcal{A} : \mathcal{X} \mapsto \mathcal{Y}$ be a linear operator. Let $L(\mathcal{X}, \mathcal{Y})$ denote the space of linear operators that map from $\mathcal{X}$ to $\mathcal{Y}$. Consider the linear system

$$\mathcal{A}x = b, \tag{6}$$

where $x \in \mathcal{X}$ and $b \in \mathbf{Range}(\mathcal{A})$. Consequently there exists a solution to the equation (6). In particular, we aim to find the solution closest to a given initial point $x_0 \in \mathcal{X}$:

$$x^* \overset{\text{def}}{=} \arg\min_{x \in \mathcal{X}} \tfrac{1}{2}\|x - x_0\|^2 \quad \text{subject to} \quad \mathcal{A}x = b. \tag{7}$$

Using the pseudoinverse and Lemma 22 item *vi*, the solution to (7) is given by

$$x^* = x_0 - \mathcal{A}^\dagger(\mathcal{A}x_0 - b) \in x_0 + \mathbf{Range}(\mathcal{A}^*), \tag{8}$$

where $A^\dagger$ and A^* denote the pseudoinverse and the adjoint of A, respectively.

3.1 The algorithm

Let $\mathcal{Z}$ be a Euclidean space and consider a random linear operator $\mathcal{S}_k \in L(\mathcal{Y}, \mathcal{Z})$ chosen from some distribution $\mathcal{D}$ over $L(\mathcal{Y}, \mathcal{Z})$ at iteration k. Our method is given in Algorithm 1, where $Z_k \in L(\mathcal{X})$ is a random linear operator given by the following compositions

$$Z_k = Z(\mathcal{S}_k) \overset{\text{def}}{=} \mathcal{A}^*\mathcal{S}_k^*(\mathcal{S}_k\mathcal{A}\mathcal{A}^*\mathcal{S}_k^*)^\dagger\mathcal{S}_k\mathcal{A}. \tag{9}$$

The updates of variables g_k and x_{k+1} on lines 8 and 9, respectively, correspond to what is known as the *sketch-and-project* update:

$$x_{k+1} = \arg\min_{x \in \mathcal{X}} \tfrac{1}{2}\|x - y_k\|^2 \quad \text{subject to} \quad \mathcal{S}_k\mathcal{A}x = \mathcal{S}_k b, \tag{10}$$

which can also be written as the following operation

$$x_{k+1} - x_* = (I - Z_k)(y_k - x_*). \tag{11}$$

This follows from the fact that $b \in \mathbf{Range}(\mathcal{A})$, together with item i of Lemma 22. Furthermore, note that the adjoint $\mathcal{A}^*$ and the pseudoinverse in Algorithm 1 are taken with respect to the norm in (7).

Algorithm 1 Accelerated Sketch-and-Project for solving (10) [24]

1: **Parameters:** $\mu, \nu > 0$, $\mathcal{D}$ = distribution over random linear operators.
2: Choose $x_0 \in \mathcal{X}$ and set $v_0 = x_0$, $\beta = 1 - \sqrt{\frac{\mu}{\nu}}$, $\gamma = \sqrt{\frac{1}{\mu\nu}}$, $\alpha = \frac{1}{1+\gamma\nu}$.
3: **for** $k = 0, 1, \ldots$ **do**
4: $\quad y_k = \alpha v_k + (1 - \alpha)x_k$
5: $\quad$ Sample an independent copy $S_k \sim \mathcal{D}$
6: $\quad g_k = \mathcal{A}^*\mathcal{S}_k^*(\mathcal{S}_k\mathcal{A}\mathcal{A}^*\mathcal{S}_k^*)^\dagger\mathcal{S}_k(\mathcal{A}y_k - b) = Z_k(y_k - x_*)$
7: $\quad x_{k+1} = y_k - g_k$
8: $\quad v_{k+1} = \beta v_k + (1 - \beta)y_k - \gamma g_k$
9: **end for**

Algorithm 1 was first proposed and analyzed by Richtárik and Takáč [24] in the special case when $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}^m$. Our contribution here is in extending the algorithm and analysis to the more abstract setting of Euclidean spaces. In addition, we provide some further extensions of this method in Sections D and E, allowing for a non-unit stepsize and variable α, respectively.

3.2 Key assumptions and quantities

Denote $Z = Z(\mathcal{S})$ for $\mathcal{S} \sim \mathcal{D}$. Assume that the *exactness property* holds

$$\mathbf{Null}(\mathcal{A}) = \mathbf{Null}(\mathbf{E}[Z]); \tag{12}$$

this is also equivalent to $\mathbf{Range}(\mathcal{A}^*) = \mathbf{Range}(\mathbf{E}[Z])$. The exactness assumption is of key importance in the sketch-and-project framework, and indeed it is not very strong. For example, it holds for the matrix inversion problem with every sketching strategy we consider. We further assume that $\mathcal{A} \neq 0$ and $\mathbf{E}[Z]$ is finite. First we collect a few observation on the Z operator

Lemma 1. *The Z operator* (9) *is a self-adjoint positive projection. Consequently $\mathbf{E}[Z]$ is a self-adjoint positive operator.*

The two parameters that govern the acceleration are

$$\mu \overset{\text{def}}{=} \inf_{x \in \mathbf{Range}(\mathcal{A}^*)} \frac{\langle \mathbf{E}[Z]x, x\rangle}{\langle x, x\rangle}, \qquad \nu \overset{\text{def}}{=} \sup_{x \in \mathbf{Range}(\mathcal{A}^*)} \frac{\langle \mathbf{E}\left[Z\mathbf{E}[Z]^\dagger Z\right]x, x\rangle}{\langle \mathbf{E}[Z]x, x\rangle}. \tag{13}$$

The supremum in the definition of ν is well defined due to the exactness assumption together with $\mathcal{A} \neq 0$.

Lemma 2. *We have*

$$1 \quad \leq \quad \nu \quad \leq \quad \tfrac{1}{\mu} \quad = \quad \|\mathbf{E}[Z]^\dagger\|. \tag{14}$$

Moreover, if $\mathbf{Range}(\mathcal{A}^*) = \mathcal{X}$, we have

$$\frac{\mathbf{Rank}(\mathcal{A}^*)}{\mathbf{E}[\mathbf{Rank}(Z)]} \leq \nu. \tag{15}$$

3.3 Convergence and change of the norm

For a positive self-adjoint $G \in L(\mathcal{X})$ and $x \in \mathcal{X}$ let $\|x\|_G \overset{\text{def}}{=} \sqrt{\langle x, x\rangle_G} \overset{\text{def}}{=} \sqrt{\langle Gx, x\rangle}$. We now informally state the convergence rate of Algorithm 1. Theorem 3 generalizes the main theorem from [24] to linear systems in Euclidean spaces.

Theorem 3. *Let x_k, v_k be the random iterates of Algorithm 1. Then*

$$\mathbf{E}\left[\|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \tfrac{1}{\mu}\|x_k - x_*\|^2\right] \leq \left(1 - \sqrt{\tfrac{\mu}{\nu}}\right)^k \mathbf{E}\left[\|v_0 - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \tfrac{1}{\mu}\|x_0 - x_*\|^2\right].$$

This theorem shows the accelerated Sketch-and-Project algorithm converges linearly with a rate of $\left(1 - \sqrt{\frac{\mu}{\nu}}\right)$, which translates to a total of $O(\sqrt{\nu/\mu}\log(1/\epsilon))$ iterations to bring the given error in Theorem 3 below $\epsilon > 0$. This is in contrast with the non-accelerated Sketch-and-Project algorithm which requires $O((1/\mu)\log(1/\epsilon))$ iterations, as shown in [12] for solving linear systems. From (14), we have the bounds $1/\sqrt{\mu} \leq \sqrt{\nu/\mu} \leq 1/\mu$. On one extreme, this inequality shows that the iteration complexity of the accelerated algorithm is at least as good as its non-accelerated counterpart. On the other extreme, the accelerated algorithm might require as little as the square root of the number of iterations of its non-accelerated counterpart. Since the cost of a single iteration of the accelerated algorithm is of the same order as the non-accelerated algorithm, this theorem shows that acceleration can offer a significant speed-up, which is verified numerically in Section 5. It is also possible to get the convergence rate of accelerated sketch-and-project where projections are taken with respect to a different weighted norm. For technical details, see Section B.4 of the Appendix.

3.4 Coordinate sketches with convenient probabilities

Let us consider a simple example in the setting for Algorithm 1 where we can understand parameters μ, ν. In particular, consider a linear system $Ax = b$ in $\mathbb{R}^n$ where A is symmetric positive definite.

Corollary 4. *Choose $B = A$ and $S = e_i$ with probability proportional to $A_{i,i}$. Then*

$$\mu = \frac{\lambda_{\min}(A)}{\mathbf{Tr}(A)} =: \mu^P \quad \textit{and} \quad \nu = \frac{\mathbf{Tr}(A)}{\min_i A_{i,i}} =: \nu^P \tag{16}$$

and therefore the convergence rate given in Theorem 3 for the accelerated algorithm is

$$\left(1 - \sqrt{\tfrac{\mu}{\nu}}\right)^k \quad = \quad \left(1 - \frac{\sqrt{\lambda_{\min}(A)\min_i A_{i,i}}}{\mathbf{Tr}(A)}\right)^k. \tag{17}$$

Rate (17) of our accelerated method is to be contrasted with the rate of the non-accelerated method: $(1-\mu)^k = (1 - \lambda_{\min}(A)/\mathbf{Tr}(A))^k$. Clearly, we gain from acceleration if the smallest diagonal element of A is significantly larger than the smallest eigenvalue.

In fact, parameters μ^P, ν^P above are the correct choice for the matrix inversion algorithm, when symmetry is not enforced, as we shall see later. Unfortunately, we are not able to estimate the parameters while enforcing symmetry for different sketching strategies. We dedicate a section in numerical experiments to test, if the parameter selection (16) performs well under enforced symmetry and different sketching strategies, and also how one might safely choose μ, ν in practice.

4 Accelerated Stochastic BFGS Update

The update of the inverse Hessian used in quasi-Newton methods (e.g., in BFGS) can be seen as a sketch-and-project update applied to the linear system $AX = I$, while $X = X^\top$ is enforced, and where A denotes and approximation of the Hessian. In this section, we present an accelerated version of these updates. We provide two different proofs: one based on Theorem 3 and one based on vectorization. By mimicking the updates of the accelerated stochastic BFGS method for inverting matrices, we determine a heuristic for accelerating the classic deterministic BFGS update. We then incorporate this acceleration into the classic BFGS optimization method and show that the resulting algorithm can offer a speed-up of the standard BFGS algorithm.

4.1 Accelerated matrix inversion

Consider the symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ and the following projection problem

$$A^{-1} = \arg\min_X \|X\|^2_{F(A)} \quad \text{subject to} \quad AX = I, \quad X = X^\top, \tag{18}$$

where $\|X\|_{F(A)} \overset{\text{def}}{=} \mathbf{Tr}\left(AX^\top AX\right) = \|A^{1/2}XA^{1/2}\|^2_F$. This projection problem can be cast as an instantiation of the general projection problem (7). Indeed, we need only note that the constraint in (18) is linear and equivalent to $\mathcal{A}(X) \overset{\text{def}}{=} \begin{pmatrix} AX \\ X - X^\top \end{pmatrix} = \begin{pmatrix} I \\ 0 \end{pmatrix}$. The matrix inversion problem can be efficiently solved using sketch-and-project with a symmetric sketch [11]. The symmetric sketch is given by $\mathcal{S}_k\mathcal{A}(X) = \begin{pmatrix} S_k^\top AX \\ X - X^\top \end{pmatrix}$, where $S_k \in \mathbb{R}^{n\times\tau}$ is a random matrix drawn from a distribution $\mathcal{D}$ and $\tau \in \mathbb{N}$. The resulting sketch-and-project method is as follows

$$X_{k+1} = \arg\min_X \|X - X_k\|^2_{F(A)} \quad \text{subject to} \quad S_k^\top AX = S_k^\top, \quad X = X^\top, \tag{19}$$

the closed form solution of which is

$$X_{k+1} = S_k(S_k^\top AS_k)^{-1}S_k^\top + \left(I - S_k(S_k^\top AS_k)^{-1}S_k^\top A\right)X_k\left(I - AS_k(S_k^\top AS_k)^{-1}S_k^\top\right). \tag{20}$$

By observing that (4.2) is the sketch-and-project algorithm applied to a linear operator equation, we have constructed an accelerated version in Algorithm 2. We can also apply Theorem 3 to prove that Algorithm 2 is indeed accelerated.

Theorem 5. *Let $L^k \overset{\text{def}}{=} \|V_k - A^{-1}\|^2_M + \frac{1}{\mu}\|X_k - A^{-1}\|^2_{F(A)}$. The iterates of Algorithm 2 satisfy*

$$\mathbf{E}\left[L_{k+1}\right] \le \left(1 - \sqrt{\tfrac{\mu}{\nu}}\right)\mathbf{E}\left[L_k\right], \tag{21}$$

where $\|X\|^2_M = \mathbf{Tr}\left(A^{1/2}X^\top A^{1/2}\mathbf{E}\left[Z\right]^\dagger A^{1/2}XA^{1/2}\right)$. Furthermore,

$$\mu \overset{\text{def}}{=} \inf_{X\in\mathbb{R}^{n\times n}} \tfrac{\langle \mathbf{E}[Z]X, X\rangle}{\langle X, X\rangle} = \lambda_{\min}(\mathbf{E}\left[\mathbf{Z}\right]), \qquad \nu \overset{\text{def}}{=} \sup_{X\in\mathbb{R}^{n\times n}} \tfrac{\langle \mathbf{E}\left[Z\mathbf{E}[Z]^\dagger Z\right]X, X\rangle}{\langle \mathbf{E}[Z]X, X\rangle}, \tag{22}$$

where

$$\mathbf{Z} \overset{\text{def}}{=} I\otimes I - (I-P)\otimes(I-P), \qquad P \overset{\text{def}}{=} A^{1/2}S(S^\top AS)^{-1}S^\top A^{1/2}, \tag{23}$$

and $Z : X \in \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is given by $Z(X) = X - (I-P)X(I-P) = XP + PX(I-P)$. Moreover, $2\lambda_{\min}(\mathbf{E}\left[P\right]) \ge \lambda_{\min}(\mathbf{E}\left[\mathbf{Z}\right]) \ge \lambda_{\min}(\mathbf{E}\left[P\right])$.

Notice that preserving symmetry yields $\mu = \lambda_{\min}(\mathbf{E}\left[\mathbf{Z}\right])$, which can be up to twice as large as $\lambda_{\min}(\mathbf{E}\left[P\right])$, which is the value of the μ parameter of the method without preserving symmetry. This improved rate is new, and was not present in the algorithm's debut publication [11]. In terms of parameter estimation, once symmetry is not preserved, we fall back onto the setting from Section 3.4. Unfortunately, we were not able to quantify the effect of enforcing symmetry on the parameter ν.

4.2 Vectorizing – a different insight

Define $\mathbf{Vec} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ to be a vectorization operator of column-wise stacking and denote $x \overset{\text{def}}{=} \mathbf{Vec}\left(X\right)$. It can be shown that the sketch-and-project operation for matrix inversion (4.2) is equivalent to

$$x_{k+1} = \arg\min_x \|x - x_k\|^2_{A\otimes A} \quad \text{subject to} \quad (I\otimes S_k^\top)(I\otimes A)x = (I\otimes S_k^\top)\mathbf{Vec}\left(I\right),\ Cx = 0,$$

Algorithm 2 Accelerated BFGS matrix inversion (solving (18))

Parameters: $\mu, \nu > 0$, $\mathcal{D}$ = distribution over random linear operators.
Choose $X_0 \in \mathcal{X}$ and set $V_0 = X_0$, $\beta = 1 - \sqrt{\frac{\mu}{\nu}}$, $\gamma = \sqrt{\frac{1}{\mu\nu}}$, $\alpha = \frac{1}{1+\gamma\nu}$
for $k = 0, 1, \ldots$ **do**
    $Y_k = \alpha V_k + (1-\alpha) X_k$
    Sample an independent copy $S \sim \mathcal{D}$
    $X_{k+1} = Y_k + (Y_k A - I) S (S^\top AS)^{-1} S^\top - S(S^\top AS)^{-1} S^\top A Y_k$
        $+ S(S^\top AS)^{-1} S^\top A Y_k A S (S^\top AS)^{-1} S^\top$
    $V_{k+1} = \beta V_k + (1-\beta) Y_k - \gamma (Y_k - X_{k+1})$
end for

where C is defined so that $Cx = 0$ if and only if $X = X^\top$. The above is a sketch-and-project update for a linear system in $\mathbb{R}^{n^2}$, which allows to obtain an alternative proof of Theorem 5, without using our results from Euclidean spaces. The details are provided in Section H.2 of the Appendix.

4.3 Accelerated BFGS as an optimization algorithm

As a tweak in the stochastic BFGS allows for a faster estimation of Hessian inverse and therefore more accurate steps of the method, one might wonder if a equivalent tweak might speed up the standard, deterministic BFGS algorithm for solving 1. The mentioned tweaked version of standard BFGS is proposed as Algorithm 3. We do not state a convergence theorem for this algorithm—due to the deterministic updates the analysis is currently elusive—nor propose to use it as a default solver, but we rather introduce it as a novel idea for accelerating optimization algorithms. We leave theoretical analysis for the future work. For now, we perform several numerical experiments, in order to understand the potential and limitations of this new method.

Algorithm 3 BFGS method with accelerated BFGS update for solving (1)

1: **Parameters:** $\mu, \nu > 0$, stepsize η.
2: Choose $X_0 \in \mathcal{X}$, w_0 and set $V_0 = X_0$, $\beta = 1 - \sqrt{\frac{\mu}{\nu}}$, $\gamma = \sqrt{\frac{1}{\mu\nu}}$, $\alpha = \frac{1}{1+\gamma\nu}$.
3: **for** $k = 0, 1, \ldots$ **do**
4: $w_{k+1} = w_k - \eta X_k \nabla f(w_k)$
5: $s_k = w_{k+1} - w_k, \quad \zeta_k = \nabla f(w_{k+1}) - \nabla f(w_k)$
6: $Y_k = \alpha V_k + (1-\alpha) X_k$
7: $X_{k+1} = \frac{\delta_k \delta_k^\top}{\delta_k^\top \zeta_k} + \left(I - \frac{\delta_k \zeta_k^\top}{\delta_k^\top \zeta_k}\right) Y_k \left(I - \frac{\zeta_k \delta_k^\top}{\delta_k^\top \zeta_k}\right)$
8: $V_{k+1} = \beta V_k + (1-\beta) Y_k - \gamma (Y_k - X_{k+1})$
9: **end for**

To better understand Algorithm 3, recall that the BFGS updates an estimate of the inverse Hessian via

$$X_{k+1} = \text{argmin}_X \ \|X - X_k\|^2_{F(A)} \quad \text{subject to} \quad X\delta_k = \zeta_k, \ X = X^\top, \tag{24}$$

where $\delta_k = w_{k+1} - w_k$ and $\zeta_k = \nabla f(w_{k+1}) - \nabla f(w_k)$. The above has the following closed form solution $X_{k+1} = \frac{\delta_k \delta_k^\top}{\delta_k^\top \zeta_k} + \left(I - \frac{\delta_k \zeta_k^\top}{\delta_k^\top \zeta_k}\right) X_k \left(I - \frac{\zeta_k \delta_k^\top}{\delta_k^\top \zeta_k}\right)$. This update appears on line 7 of Algorithm 3 with the difference being that it is applied to a matrix Y_k.

5 Numerical Experiments

We perform extensive numerical experiments to bring additional insight to both the performance of and to parameter selection for Algorithms 2 and 3. More numerical experiments can be found in Section A of the appendix. We first test our accelerated matrix inversion algorithm, and subsequently perform experiments related to Section 4.3.

5.1 Accelerated Matrix Inversion

We consider the problem of inverting a matrix symmetric positive matrix A. We focus on a few particular choices of matrices A (specified when describing each experiment), that differ in their eigenvalue spectra. Three different sketching strategies are studied: Coordinate sketches with convenient probabilities ($S = e_i$ with probability proportional to $A_{i,i}$), coordinate sketches with uniform probabilities ($S = e_i$ with probability $\frac{1}{n}$) and Gaussian sketches ($S \sim \mathcal{N}(0, I)$). As matrices to be inverted, we use both artificially generated matrices with the access to the spectrum and also Hessians of ridge regression problems from LIBSVM.

We have shown earlier that μ, ν can be estimated as per (16) for coordinate sketches with convenient probabilities without enforcing symmetry. We use the mentioned parameters for the other sketching strategies while enforcing the symmetry. Since in practice one might not have an access to the exact parameters μ, ν for given sketching strategy, we test sensitivity of the algorithm to parameter choice . We also test test for ν chosen by (16), $\mu = \frac{1}{100\nu}$ and $\mu = \frac{1}{10000\nu}$.

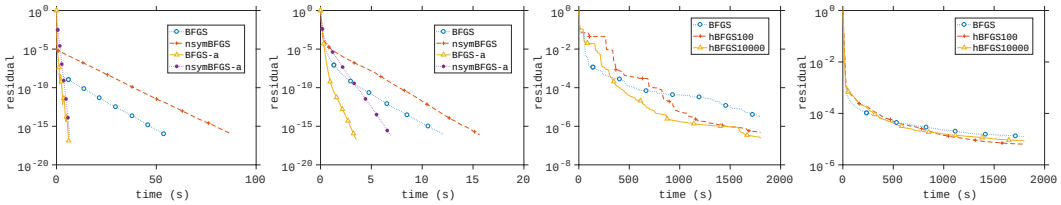

Figure 1: From left to right: (i) Eigenvalues of $A \in \mathbb{R}^{100\times 100}$ are $1, 10^3, 10^3, \ldots, 10^3$ and coordinate sketches with convenient probabilities are used. (ii) Eigenvalues of $A \in \mathbb{R}^{100\times 100}$ are $1, 2, \ldots, n$ and Gaussian sketches are used. Label "nsym" indicates non-enforcing symmetry and "-a" indicates acceleration. (iii) Epsilon dataset ($n = 2000$), coordinate sketches with uniform probabilities. (iv) SVHN dataset ($n = 3072$), coordinate sketches with convenient probabilities. Label "h" indicates that $\lambda_{\min}$ was not precomputed, but μ was chosen as described in the text.

For more plots, see Section A in the appendix as here we provide only a tiny fraction of all plots. The experiments suggest that once the parameters μ, ν are estimated exactly, we get a speedup comparing to the nonaccelerated method; and the amount of speedup depends on the structure of A and the sketching strategy. We observe from Figure 1 that we gain a great speedup for ill conditioned problems once the eigenvalues are concentrated around the largest eigenvalue. We also observe from Figure 1 that enforcing symmetry combines well with μ, ν computed for the algorithm which do not enforce symmetry. On top of that, choice of μ, ν per (16) seems to be robust to different sketching strategies, and in worst case performs as fast as nonaccelerated algorithm.

5.2 BFGS Optimization Method

We test Algorithm 3 on several logistic regression problems using data from LIBSVM [5]. In all our tests we centered and normalized the data, included a bias term (a linear intercept), and choose the regularization parameter as $\lambda = 1/m$, where m is the number of data points. To keep things as simple as possible, we also used a fixed stepsize which was determined using grid search. Since our theory regarding the choice for the parameters μ and ν does not apply in this setting, we simply probed the space of parameters manually and reported the best found result, see Figure 2. In the legend we use BFGS-a-μ-ν to denote the accelerated BFGS method (Alg 3) with parameters μ and ν.

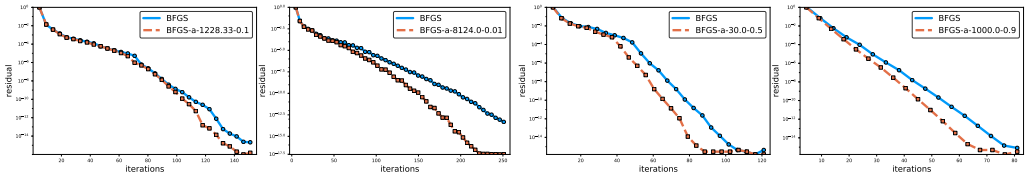

Figure 2: Algorithm 3 (BFGS with accelerated matrix inversion quasi-Newton update) vs standard BFGS. From left to right: `phishing`, `mushrooms`, `australian` and `splice` dataset.

On all four datasets, our method outperforms the classic BFGS method, indicating that replacing classic BFGS update rules for learning the inverse Hessian by our new accelerated rules can be beneficial in practice. In A.4 in the appendix we also show the time plots for solving the problems in Figure 2, and show that the accelerated BFGS method also converges faster in time.

References

[1] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *The Journal of Machine Learning Research*, 18(1):4148–4187, 2017.

[2] Albert S. Berahas, Raghu Bollapragada, and Jorge Nocedal. An investigation of Newton-sketch and subsampled Newton methods. *CoRR*, abs/1705.06211, 2017.

[3] Charles G Broyden. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

[4] Richard H. Byrd, S. L. Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[5] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[6] C. A. Desoer and B. H. Whalen. A note on pseudoinverses. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):442–447, 1963.

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

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

[9] Robert Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *International Conference on Machine Learning*, pages 1869–1878, 2016.

[10] Robert M. Gower and Peter Richtárik. Stochastic dual ascent for solving linear systems. *arXiv:1512.06890*, 2015.

[11] Robert M. Gower and Peter Richtárik. Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1380–1409, 2017.

[12] Robert Mansel Gower and Peter Richtárik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

[13] S. Kaczmarz. Angenäherte Auflösung von Systemen linearer Gleichungen. *Bulletin International de l'Académie Polonaise des Sciences et des Lettres*, 35:355–357, 1937.

[14] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[15] Ji Liu and Stephen J. Wright. An accelerated randomized Kaczmarz algorithm. *Math. Comput.*, 85(297):153–178, 2016.

[16] Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, Newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*, 2017.

[17] Aryan Mokhtari and Alejandro Ribeiro. Global convergence of online limited memory BFGS. *The Journal of Machine Learning Research*, 16:3151–3181, 2015.

[18] Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pages 249–258, 2016.

[19] Yu. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[20] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[21] Yurii Nesterov and Sebastian U. Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.

[22] G.K. Pedersen. *Analysis Now*. Graduate Texts in Mathematics. Springer New York, 1996.

[23] Mert Pilanci and Martin J. Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

[24] Peter Richtárik and Martin Takáč. Stochastic reformulations of linear systems: accelerated method. *Manuscript, October 2017*, 2017.

[25] Peter Richtárik and Martin Takáč. Stochastic reformulations of linear systems: algorithms and convergence theory. *arXiv:1706.01108*, 2017.

[26] Nicol N Schraudolph and G Simon. A stochastic quasi-Newton method for online convex optimization. *In Proceedings of 11th International Conference on Artificial Intelligence and Statistics*, 2007.

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

[28] S. U. Stich, C. L. Müller, and B. Gärtner. Variable metric random pursuit. *Mathematical Programming*, 156(1):549–579, Mar 2016.

[29] Sebastian U. Stich. *Convex Optimization with Random Pursuit*. PhD thesis, ETH Zurich, 2014. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 22111.

[30] Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

[31] Stephen Tu, Shivaram Venkataraman, Ashia C. Wilson, Alex Gittens, Michael I. Jordan, and Benjamin Recht. Breaking locality accelerates block Gauss-Seidel. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3482–3491, 2017.

[32] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[33] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-Newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[34] Stephen J. Wright. Coordinate descent algorithms. *Math. Program.*, 151(1):3–34, June 2015.

[35] Peng Xu, Jiyan Yang, Farbod Roosta-Khorasani, Christopher Ré, and Michael W Mahoney. Sub-sampled newton methods with non-uniform sampling. In *Advances in Neural Information Processing Systems*, pages 3000–3008, 2016.

A Further Experiments with Accelerated quasi-Newton Updates

In this section, we test the the empirical rate of convergence of Algorithm 2, the accelerated BFGS update for inverting positive definite matrices. Only vector sketches are considered, as the standard quasi-Newton methods also update the inverse Hessian only according to the action in one direction. We compare the speed of the accelerated method with precomputed estimates of the parameters μ, ν to the nonaccelerated method. The precomputed estimates of μ^P, ν^P are set as per (16):

$$\mu^P = \frac{\lambda_{\min}(A)}{\mathbf{Tr}\,(A)}, \qquad \nu^P = \frac{\mathbf{Tr}\,(A)}{\min_i(A_{i,i})},$$

which is the optimal choice for coordinate sketches with convenient probabilities without enforcing symmetry. In practice we might not have an access to $\lambda_{\min}(A)$, thus we cannot compute μ^P exactly. Therefore we also test sensitivity of the algorithm to the choice of parameters, and we run some experiments where we only guess parameter μ^P.

Lastly, the tests are performed on both artificial examples and LIBSVM [5] data. We shall also explain the legend of plots: "a" indicates acceleration, "nsym" indicates the algorithm without enforcing symmetry and "h" indicates the setting when ν^P is not known, and a naive heuristic choice is casted.

A.1 Simple and well understood artificial example

Let us consider inverting the matrix $A = \alpha I + \beta \mathbf{1}\mathbf{1}^\top$ for $\alpha > 0$ and $\beta \geq -\frac{\alpha}{n}$ so as in this case we have control over both μ and ν. This artificial example was considered in [31] for solving linear systems. In particular, we show that for coordinate sketches with convenient probabilities (which is indeed the same as uniform probabilities in this example), we have

$$\begin{aligned}
\mu^P &\overset{\text{def}}{=} \lambda_{\min}(\mathbf{E}\,[P]) = \frac{\min\left(\alpha, \alpha + n\beta\right)}{n(\alpha+\beta)}, \\
\nu^P &\overset{\text{def}}{=} \lambda_{\max}\left(\mathbf{E}\left[\mathbf{E}\,[P]^{-\frac{1}{2}} P \mathbf{E}\,[P]^{-1} P \mathbf{E}\,[P]^{-\frac{1}{2}}\right]\right) = n.
\end{aligned}$$

Due to the fact that we do not have a theoretical justification of μ, ν for $n > 2$ when enforcing symmetry, we set $\mu = \mu^P$ and $\nu = \nu^P$ for Gaussian sketches as well.

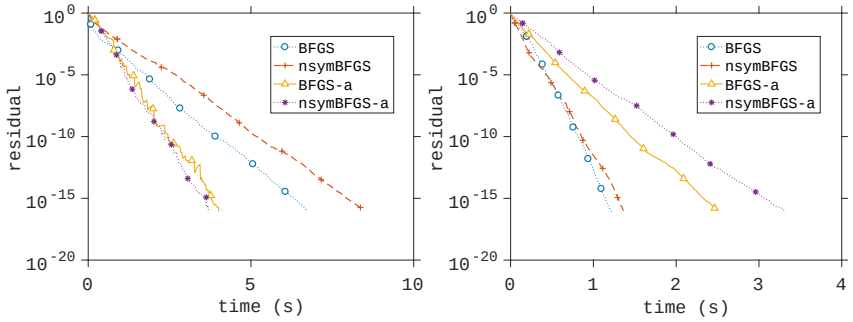

Figure 3: Parameter choice: $\alpha = 1 + 10^{-1}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch respectively.

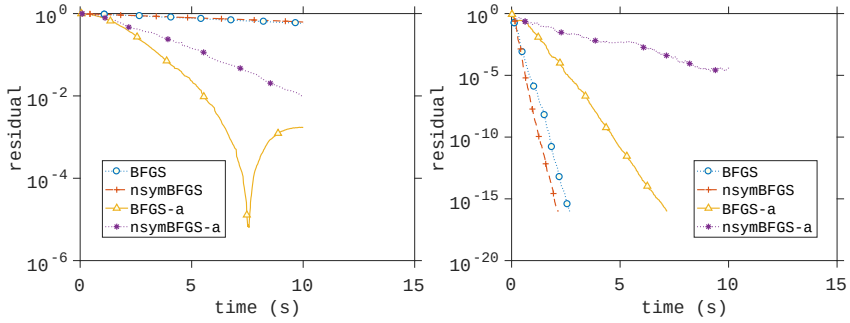

Figure 4: Parameter choice: $\alpha = 1 + 10^{-3}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch respectively.

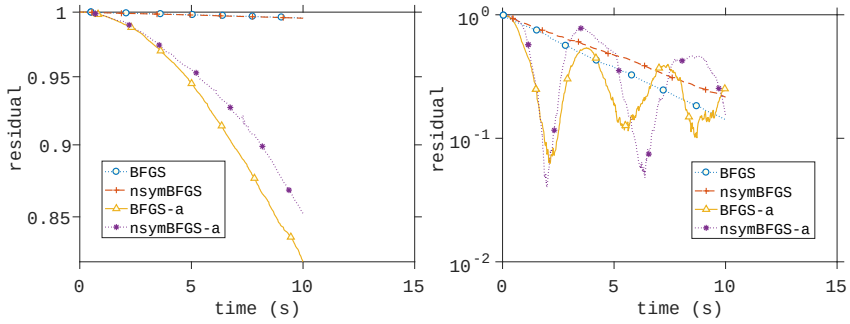

Figure 5: Parameter choice: $\alpha = 1 + 10^{-5}, \beta = -n^{-1}, n = 100$. From left to right we have: Coordinate sketch with uniform (convenient) probabilities and Gaussian sketch, respectively.

As expected from the theory, as the matrix to be inverted becomes more ill conditioned, the accelerated method performs significantly better compared to the nonaccelerated method for coordinate sketches. In fact, an arbitrary speedup can be obtained by setting $\beta = -n^{-1}$ and $\alpha \to 1$ for the coordinate sketches setup. On the other hand, Gaussian sketches report the slowing of the algorithm, most likely caused by the fact that the theoretical parameters μ, ν for Gaussian sketches with enforced symmetry are different to μ^P, ν^P, which are estimated for coordinate sketches without enforced symmetry. In the case of coordinate sketches with symmetry enforced, we suspect a great speedup even though the parameters μ, ν were set to μ^P, ν^P.

A.2 Random artificial example

We randomly generate an orthonormal matrix U, choose diagonal matrix D, and set $A = UDU^\top$. Clearly, diagonal elements of D are eigenvalues of A. We set them in the following way:

- Uniform grid. The eigenvalues are set to $1, 2, \ldots, n$.
- One small, the rest larger. The smallest eigenvalue is 1, remaining eigenvalues are all 10 in the first example, all 100 in the second example and all 1000 in the third example in this category.
- One large, the rest small. The largest eigenvalue is 10^4, the remaining eigenvalues are all 1.

Firstly, consider coordinate sketches with convenient probabilities. Notice that we can easily estimate ν^P, μ^P due to the results from Section 3.4 since we have control of $\lambda_{\min}(A)$ and therefore also of μ. Therefore, we set $\mu = \mu^P = \min D_{i,i}$ and $\nu = \nu^P$ for Algorithm 2. Then, we consider coordinate sketches with uniform probabilities and Gaussian sketches. In both cases, we set the parameters μ, ν as for coordinate sketches with convenient probabilities.

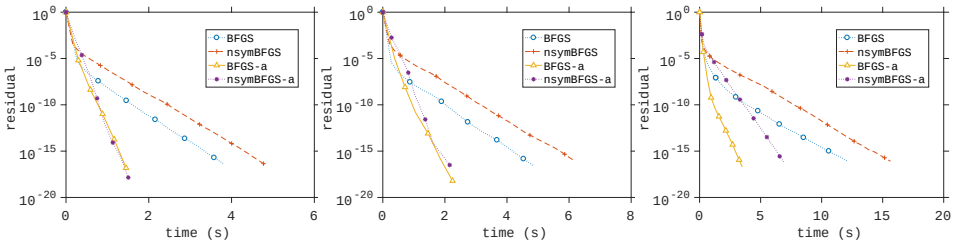

Figure 6: Eigenvalues set to $1, 2, 3, \ldots n$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

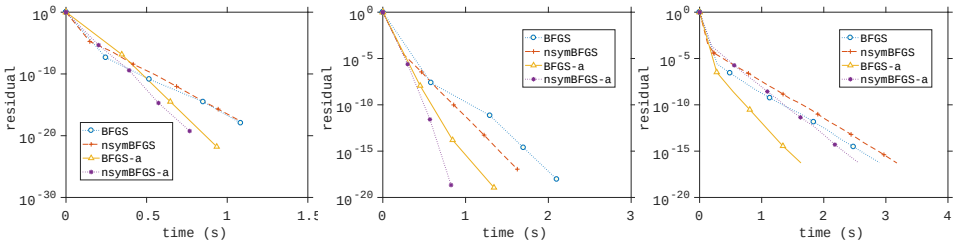

Figure 7: Eigenvalues set to $1, 10, 10, \ldots 10$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

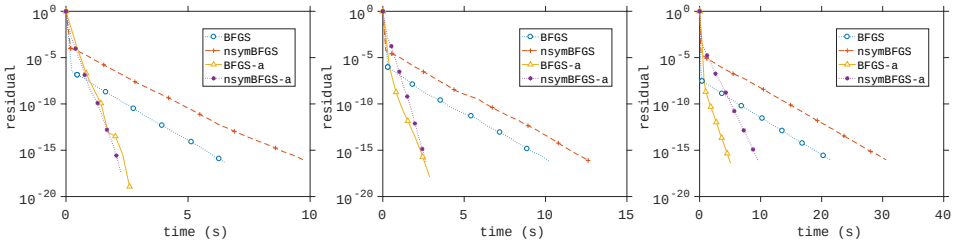

Figure 8: Eigenvalues set to $1, 100, 100, \ldots 100$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

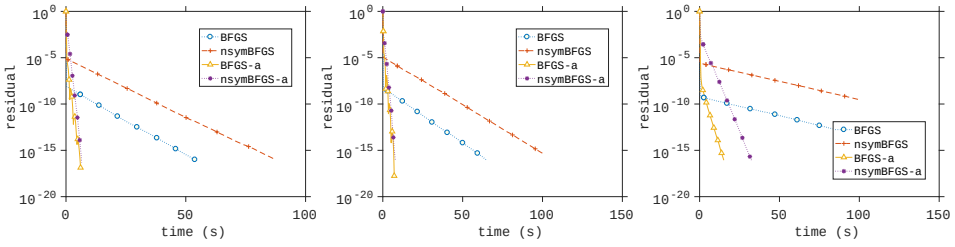

Figure 9: Eigenvalues set to $1, 1000, 1000, \ldots 1000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

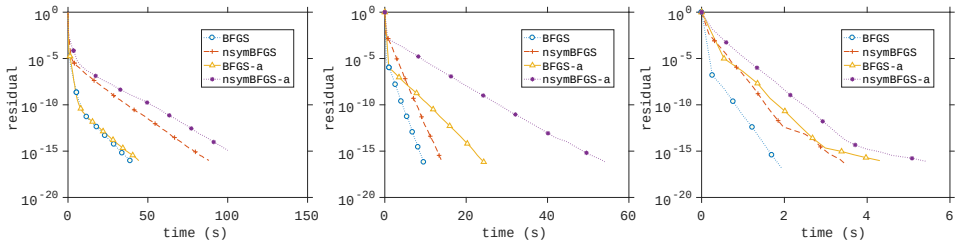

Figure 10: Eigenvalues set to $10000, 1, 1, \ldots 1$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

The numerical experiments in this section indicate that one might choose μ, ν as per Section 3.4. In other words, one might pretend to be in the setting when symmetry is not enforced and coordinate sketches with convenient probabilities are used. In fact, the practical speedup coming from the acceleration depends very strongly on the structure of matrix A. Another message to be delivered is that both preserving symmetry and acceleration yield a better convergence and they combine together well.

We also consider a problem where we pretend to not have access to $\lambda_{\min}(A)$, therefore we cannot choose $\mu = \mu^P$. Instead, we naively choose $\mu = \frac{1}{100\nu}$ and $\mu = \frac{1}{10000\nu}$.

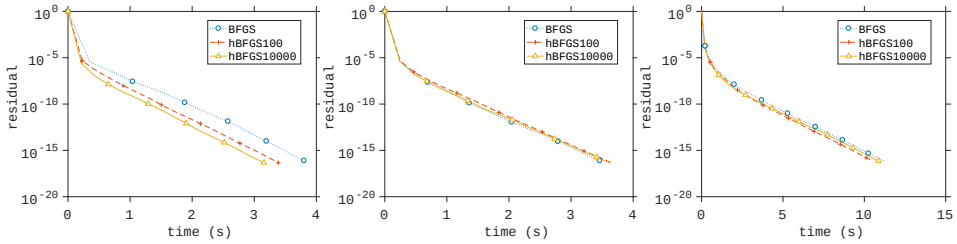

Figure 11: Eigenvalues set to $1, 2, \ldots, n$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

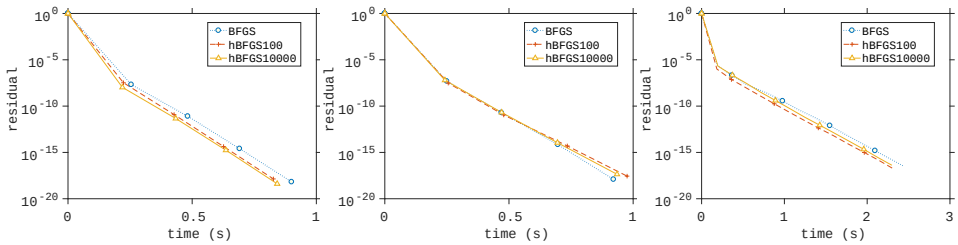

Figure 12: Eigenvalues set to $1, 10, 10, \ldots 10$. Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

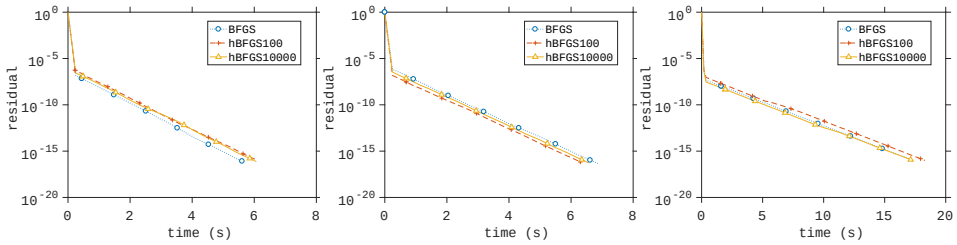

Figure 13: Eigenvalues set to $1, 100, 100, \ldots 100$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

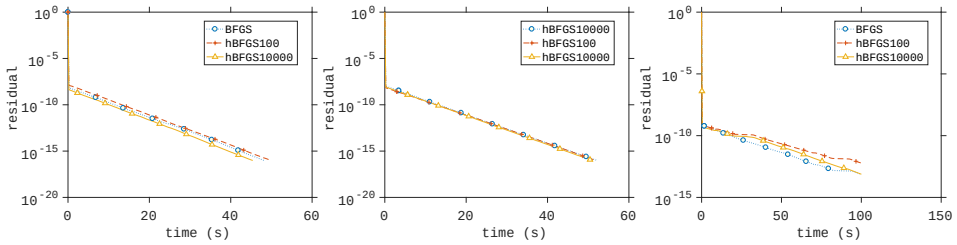

Figure 14: Eigenvalues set to $1, 1000, 1000, \ldots 1000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

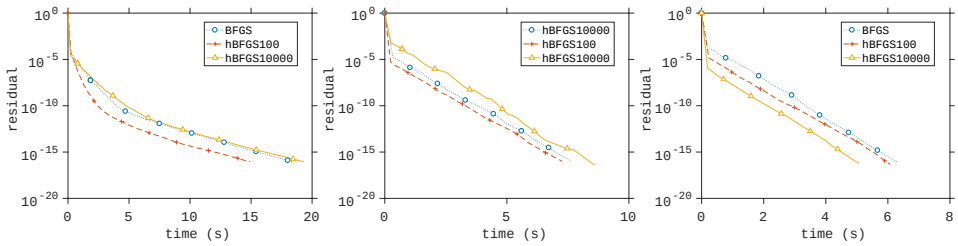

Figure 15: Eigenvalues set to $10000, 1, 1, \ldots 1$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

Notice that once the acceleration parameters are not set exactly (but they are still reasonable), we observe that the performance of the accelerated algorithm is essentially the same as the performance of the nonaccelerated algorithm. We have observed the similar behavior when setting $\mu = \mu^P$ for Gaussian sketches.

A.2.1 Sensitivity to the acceleration parameters

Here we investigate the sensitivity of the accelerated BFGS to the parameters μ and ν. First we compute ν^P, μ^P and from this we extract the following exponential grids: $\mu_i = 2^{i-4}\mu$ and $\nu_i = 5^{i-4}\nu$ for $i = 1, 2, \ldots 7$. To gauge the gain is using acceleration with a particular (μ, ν) pair, we run the accelerated algorithm for a fixed time then store the error of the final iterate. We then compute average per iteration decrease and divide it by average per iteration decrease of nonaccelerated algorithm. Thus if the resulting difference is less than one, then the accelerated algorithm was faster to nonaccelerated.

In the plots below, $n = 200$ was chosen. We focused on 2 problems described in the previous section—when the eigenvalues are uniformly distributed and when the the largest eigenvalue have multiplicity $n - 1$.

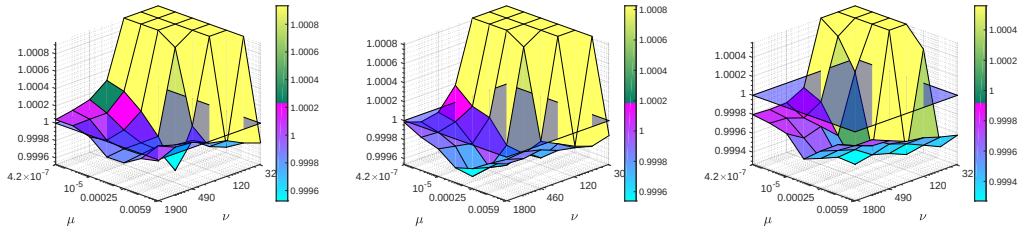

Figure 16: Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 2 \ldots, n$. From left to right we have: Coordinate sketches with convenient probabilities, coordiante sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (16) in the middle of plots. Each instance was run for 5 seconds.

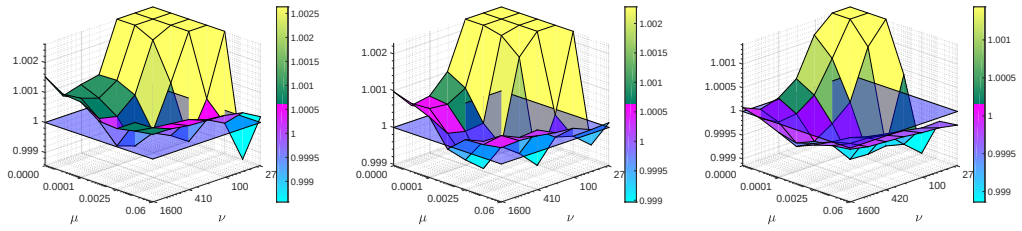

Figure 17: Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 10, 10, \ldots, 10$. From left to right we have: Coordinate sketches with convenient probabilities, coordiante sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (16) in the middle of plots. Each instance was run for 2 seconds.

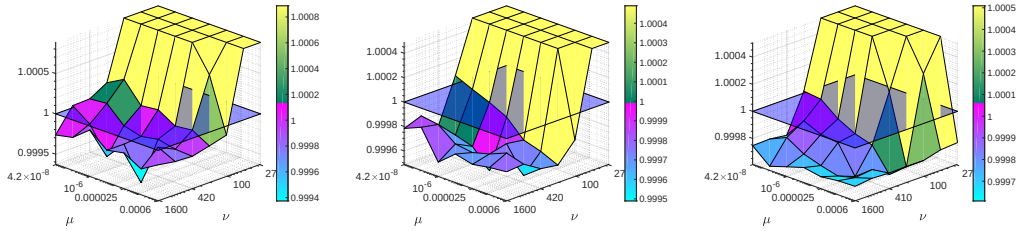

Figure 18: Sensitivity to acceleration parameters. Eigenvalues of A are set to $1, 1000, 1000, \ldots, 1000$. From left to right we have: Coordinate sketches with convenient probabilities, coordiante sketches with uniform probabilities and Gaussian sketches. Choice of parameters as per (16) in the middle of plots. Each instance was run for 10 seconds.

The crucial aspect to make the accelerated algorithm to converge is to set ν large enough. In fact, combination of both small ν and small μ leads almost always to non-convergent algorithm. On the other hand, it seems that once ν is chosen correctly, big enough μ leads to fast convergence. This indicates how to compute μ in practice (recall that computing ν is feasible)—one needs just to choose it small enough (definitely smaller than $\frac{1}{\nu}$).

A.3 Experiments with LIBSVM

Next we investigate if the accelerated BFGS update improves upon the standard BFGS update when applied to the Hessian $\nabla^2 f(x)$ of ridge regression problems of the form

$$\min_{x\in\mathbb{R}^n} f(x) \stackrel{\text{def}}{=} \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2, \qquad \nabla^2 f(x) = A^\top A + \lambda I, \tag{25}$$

using data from LIBSVM [5]. Datapoints (rows of A) were normalized such that $\|A_{i:}\|^2 = 1$ for all i and the regularization parameter was chosen as $\lambda = \frac{1}{m}$.

First, we run the experiments on smaller problems when parameters μ, ν are precomputed for coordinate sketches with convenient probabilities (16).

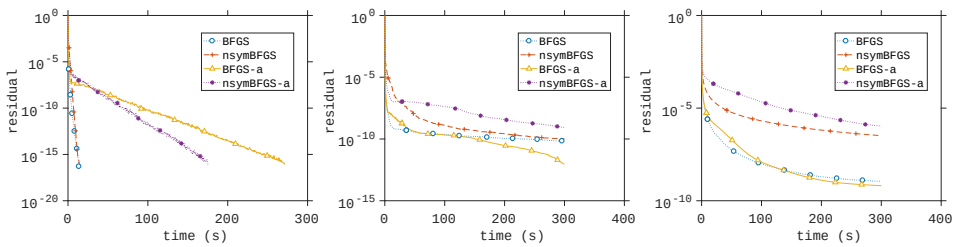

Figure 19: Dataset aloi: $n = 128$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

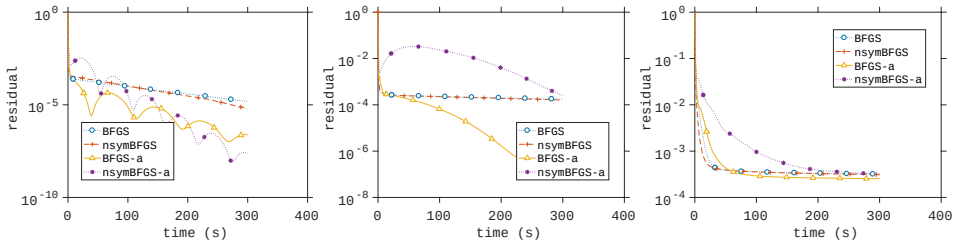

Figure 20: Dataset w1a: $n = 300$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

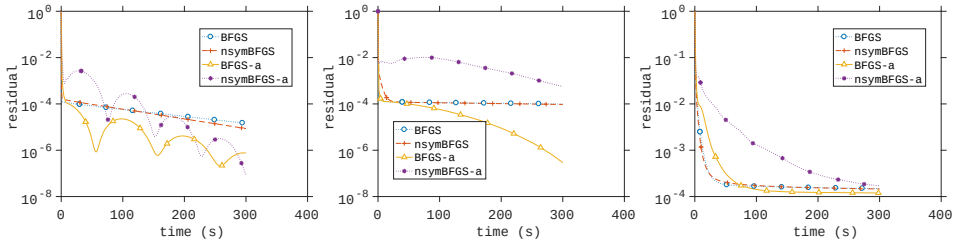

Figure 21: Dataset w2a: $n = 300$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

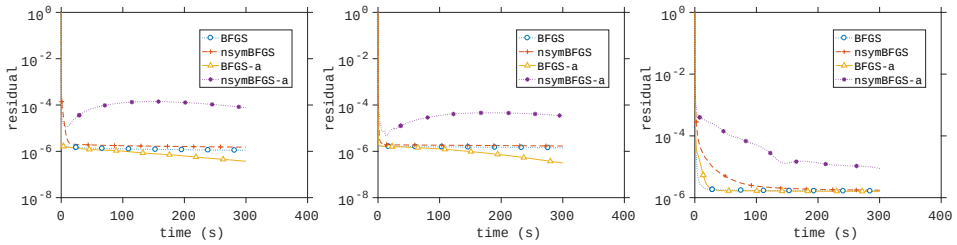

Figure 22: Dataset mushrooms: $n = 112$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

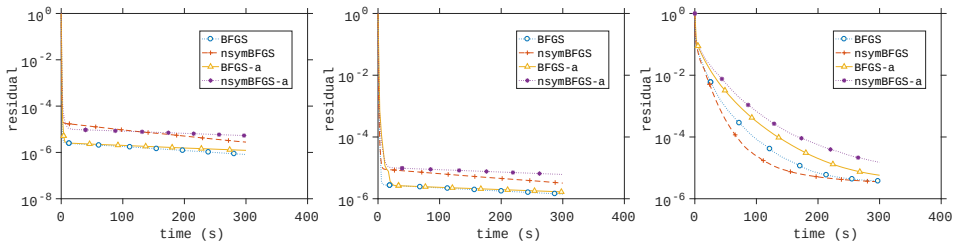

Figure 23: Dataset protein: $n = 357$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

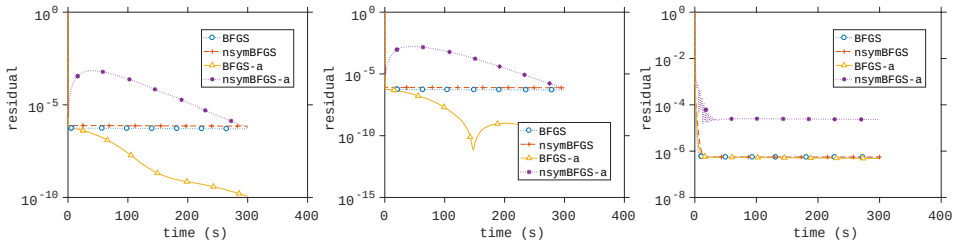

Figure 24: Dataset phishing: $n = 68$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

In the vast majority of examples, the accelerated method performed significantly better than the nonaccelerated method for coordinate sketches (with both convenient and uniform probabilities), however the methods were comparable for Gaussian sketches. We believe that this is due to the fact that choice of parameters as per (16) is close to the optimal parameters for coordinate sketches, and further for Gaussian sketches. However, the experiments on coordinate sketches indicates that for some classes of problems, accelerated algorithms with finely tuned parameters bring a great speedup compared to nonaccelerated ones.

We also consider a problem where we do not compute $\lambda_{\min}(A)$, and therefore we cannot choose $\mu = \mu^P$ in (16). Instead, we choose $\mu = \frac{1}{100\nu}$ and $\mu = \frac{1}{10000\nu}$.

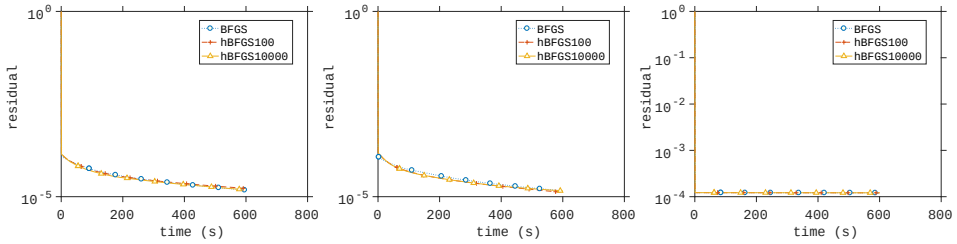

Figure 25: Dataset madelon: $n = 500$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

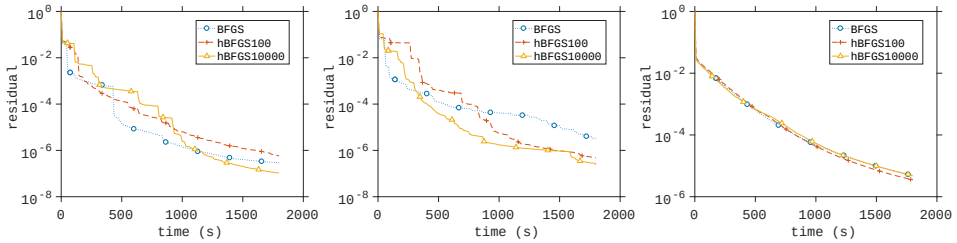

Figure 26: Dataset epsilon: $n = 2000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

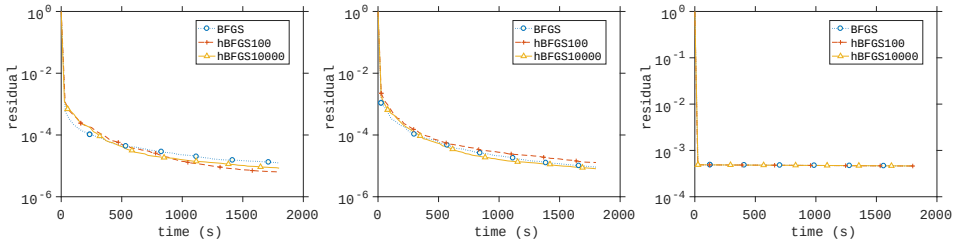

Figure 27: Dataset svhn: $n = 3072$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

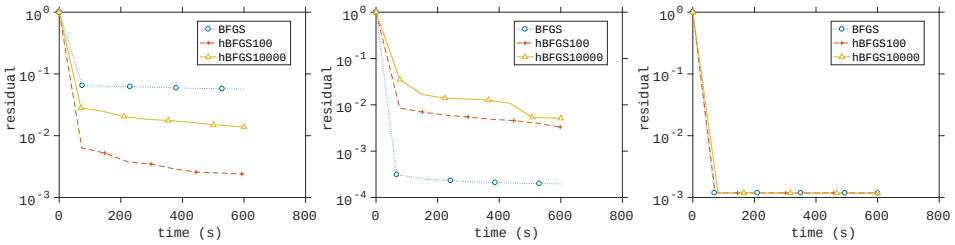

Figure 28: Dataset gisette: $n = 5000$. From left to right we have: Coordinate sketch with convenient probabilities, coordinate sketch with uniform probabilities and Gaussian sketch respectively.

Notice that once the acceleration parameters are not set exactly (but they are still reasonable), we observe that the performance of the accelerated algorithm is essentially the same as the performance of the nonaccelerated algorithm, which is essentially the same conclusion as for artificially generated examples.

A.4 Additional optimization experiments

In Figure 29 we solve the same problems with the same setup as in 29, but now we plot the time versus the residual (as opposed to iterations versus the residual). Despite the more costly iterations, the accelerated BFGS method can still converge faster than the classic BFGS method.

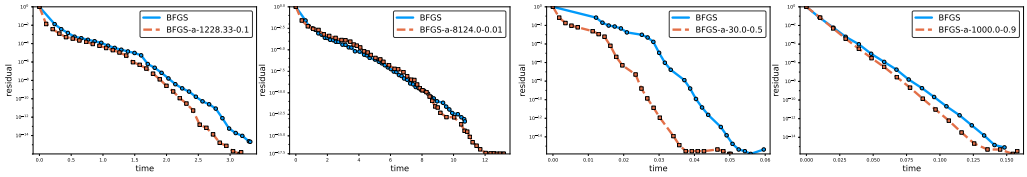

Figure 29: Algorithm 3 (BFGS with accelerated matrix inversion quasi-Newton update) vs standard BFGS. From left to right: `phishing`, `mushrooms`, `australian` and `splice` dataset.

We also give additional experiments with the same setup to the ones found in Section 5.2. Much like the `phishing` problem in Figure 2, the problems `madelon`, `covtype` and `a9a` in Figures 30, 31 and 32 did not benefit that much from acceleration. Indeed, we found in our experiments that even when choosing extreme values of μ and ν, the generated inverse Hessian would not significantly deviate from the estimate that one would obtain using the standard BFGS update. Thus on these two problems there is apparently little room for improvement by using acceleration.

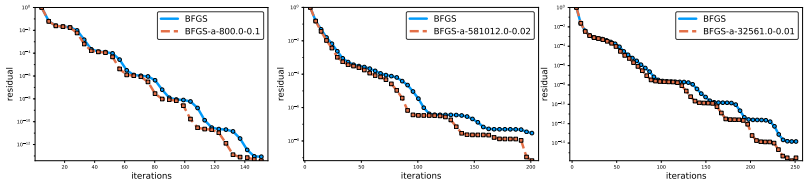

Figure 30: `madelon`: Figure 31: `covtype` Figure 32: `a9a`

B Proofs for Section 3

B.1 Proof of Lemma 2

First note that Z is a self-adjoint positive operator and thus so is $\mathbf{E}[Z]$. Consequently.

$$\begin{aligned}
\mu &\overset{(13)}{=} \inf_{x\in\mathbf{Range}(\mathcal{A}^*)} \frac{\langle \mathbf{E}[Z]\, x, x\rangle}{\langle x, x\rangle} \\
&\overset{(12)}{=} \inf_{x\in\mathbf{Range}(\mathbf{E}[Z])} \frac{\langle \mathbf{E}[Z]\, x, x\rangle}{\langle x, x\rangle} \\
&\overset{\text{Lemma 22 item ii}}{=} \inf_{x\in\mathcal{X}} \frac{\langle \mathbf{E}[Z]\, \mathbf{E}[Z]^\dagger\, x, \mathbf{E}[Z]^\dagger\, x\rangle}{\langle \mathbf{E}[Z]^\dagger\, x, \mathbf{E}[Z]^\dagger\, x\rangle} \\
&\overset{\text{Lemma 22 item i}}{=} \inf_{x\in\mathcal{X}} \frac{\langle \mathbf{E}[Z]^\dagger\, x, x\rangle}{\langle \mathbf{E}[Z]^\dagger\, x, \mathbf{E}[Z]^\dagger\, x\rangle} \\
&\overset{\text{Lemma 18}}{=} \inf_{z\in\mathbf{Range}\left((\mathbf{E}[Z]^\dagger)^{1/2}\right)} \frac{\langle z, z\rangle}{\langle \mathbf{E}[Z]^\dagger\, z, z\rangle} \qquad (\text{set } z = (\mathbf{E}[Z]^\dagger)^{1/2}x) \\
&\overset{(71)}{=} \frac{1}{\|\mathbf{E}[Z]^\dagger\|}.
\end{aligned} \tag{26}$$

For the bounds (14) we have that

$$\begin{aligned}
\nu &\overset{(13)}{=} \sup_{x\in\mathbf{Range}(\mathcal{A}^*)} \frac{\mathbf{E}\left[\langle \mathbf{E}[Z]^\dagger\, Zx, Zx\rangle\right]}{\langle \mathbf{E}[Z]\, x, x\rangle} \\
&\leq \sup_{x\in\mathbf{Range}(\mathcal{A}^*)} \frac{\|\mathbf{E}[Z]^\dagger\| \mathbf{E}\left[\|Zx\|_2^2\right]}{\langle \mathbf{E}[Z]\, x, x\rangle} \\
&= \|\mathbf{E}[Z]^\dagger\| \\
&\overset{(26)}{\leq} \frac{1}{\mu}.
\end{aligned}$$

To bound ν from below we use that $\mathbf{E}[Z]^\dagger$ is self adjoint together with that the map $X \mapsto \langle X\mathbf{E}[Z]^\dagger\, Xx, x\rangle$ is convex over the space of self-adjoint operators $X \in L(\mathcal{X})$ and for a fixed $x \in \mathcal{X}$. Consequently by Jensen's inequality

$$\mathbf{E}\left[\langle Z\mathbf{E}[Z]^\dagger\, Zx, x\rangle\right] \geq \langle \mathbf{E}[Z]\, \mathbf{E}[Z]^\dagger\, \mathbf{E}[Z]\, x, x\rangle \overset{\text{Lemma 22 item i}}{=} \langle \mathbf{E}[Z]\, x, x\rangle. \tag{27}$$

Finally

$$\nu \overset{(27)}{\geq} \sup_{x\in\mathbf{Range}(\mathcal{A}^*)} \frac{\langle \mathbf{E}[Z]\, x, x\rangle}{\langle \mathbf{E}[Z]\, x, x\rangle} = 1.$$

Lastly, to show (15) we have

$$\begin{aligned}
\mathbf{Rank}(\mathcal{A}^*) &\overset{(12)}{=} \mathbf{Rank}(\mathbf{E}[Z]) \\
&\overset{\text{Lemma 17+ Lemma 22 } (v)}{=} \mathbf{Tr}\left(\mathbf{E}[Z]\, \mathbf{E}[Z]^\dagger\right) = \mathbf{E}\left[\mathbf{Tr}\left(Z\mathbf{E}[Z]^\dagger\right)\right] \\
&= \mathbf{E}\left[\mathbf{Tr}\left(Z\mathbf{E}[Z]^\dagger\, Z\right)\right] \\
&\leq \nu\mathbf{E}[\mathbf{Tr}(Z)] \overset{\text{Lemma 17}}{=} \nu\mathbf{E}[\mathbf{Rank}(Z)],
\end{aligned}$$

where we used that $\langle \mathbf{E}\left[Z\mathbf{E}[Z]^\dagger\, Z\right] u, u\rangle \leq \nu\langle \mathbf{E}[Z]\, u, u\rangle$ for every $u \in \mathbf{Range}(\mathbf{E}[Z]) = \mathbf{Range}(\mathcal{A}^*) = \mathcal{X}$. □

Proof that $X \mapsto \langle X\mathbf{E}\left[Z\right]^{\dagger} Xx, x\rangle = \|Xx\|^2_{\mathbf{E}[Z]^{\dagger}}$ is convex: Let $G = \mathbf{E}\left[Z\right]^{\dagger}$ then

$$\begin{aligned}\|(\lambda X + (1-\lambda)Y)x\|_G^2 &= \lambda^2\|Xx\|_G^2 + (1-\lambda)^2\|Yx\|_G^2 + 2\lambda(1-\lambda)\langle xXGY, x\rangle \\ &= -\lambda(1-\lambda)\|(X-Y)x\|_G^2 \\ &\quad +\lambda\|Xx\|_G^2 + (1-\lambda)\|Yx\|_G^2 \\ &\leq \lambda\|Xx\|_G^2 + (1-\lambda)\|Yx\|_G^2.\end{aligned}$$

□

B.2 Technical lemmas to prove Theorem 3

Lemma 6. *For all $k \geq 0$, the vectors $y_k - x_*$, $x_k - x_*$ and $v_k - x_*$ belong to* $\mathbf{Range}\left(\mathcal{A}^*\right)$.

Proof. Note that $x_0 = y_0 = x_0$ and in view of (8) we have $x_* \in x_0 + \mathbf{Range}\left(\mathcal{A}^*\right)$. So $y_0 - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$, $v_0 - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$ and $x_0 - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$. Assume by induction that $y_k - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$, $v_k - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$ and $x_k - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$. Since $g_k \in \mathbf{Range}\left(\mathcal{A}^*\right)$ and $x_{k+1} = y_k - g_k$ we have

$$x_{k+1} - x_* = (y_k - x_*) - g_k \in \mathbf{Range}\left(\mathcal{A}^*\right).$$

Moreover,

$$v_{k+1} - x_* = \beta(v_k - x_*) + (1-\beta)(y_k - x*) - \gamma g_k \in \mathbf{Range}\left(\mathcal{A}^*\right).$$

Finally

$$y_{k+1} - x_* = \alpha v_{k+1} + (1-\alpha)x_{k+1} - x_* = \alpha(v_{k+1} - x_*) + (1-\alpha)(x_{k+1} - x_*) \in \mathbf{Range}\left(\mathcal{A}^*\right).$$

□

Lemma 7.

$$\mathbf{E}\left[\|Z_k(y_k - x_*)\|^2_{\mathbf{E}[Z]^{\dagger}} \mid y_k\right] \leq \nu\|y_k - x_*\|^2_{\mathbf{E}[Z]} \tag{28}$$

Proof. Since $y_k - x_* \in \mathbf{Range}\left(\mathcal{A}^*\right)$ we have that

$$\begin{aligned}\mathbf{E}\left[\|Z_k(y_k - x_*)\|^2_{\mathbf{E}[Z]^{\dagger}} \mid y_k\right] &= \langle \mathbf{E}\left[Z_k\mathbf{E}\left[Z\right]^{\dagger} Z_k\right](y_k - x_*), (y_k - x_*)\rangle \\ &\overset{(13)}{\leq} \nu\langle \mathbf{E}\left[Z\right](y_k - x_*), (y_k - x_*)\rangle \\ &= \nu\|y_k - x_*\|^2_{\mathbf{E}[Z]}.\end{aligned}$$

□

Lemma 8.

$$\|y_k - x_*\|^2_{\mathbf{E}[Z]} = \|y_k - x_*\|^2 - \mathbf{E}\left[\|x_{k+1} - x_*\|^2 \mid y_k\right] \tag{29}$$

Proof.

$$\begin{aligned}\mathbf{E}\left[\|x_{k+1} - x_*\|^2 \mid y_k\right] &= \mathbf{E}\left[\|(I - Z_k)(y_k - x_*)\|^2 \mid y_k\right] \\ &= \langle (I - \mathbf{E}\left[Z\right])(y_k - x_*), y_k - x_*\rangle \\ &= \|y_k - x_*\|^2 - \|y_k - x_*\|^2_{\mathbf{E}[Z]}.\end{aligned}$$

□

B.3 Proof of Theorem 3

Let $r_k \overset{\text{def}}{=} \|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger}$. It follows that

$$\begin{aligned} r_{k+1}^2 &= \|v_{k+1} - x_*\|^2_{\mathbf{E}[Z]^\dagger} \\ &= \|\beta v_k + (1-\beta)y_k - x_* - \gamma Z_k(y_k - x_*)\|^2_{\mathbf{E}[Z]^\dagger} \\ &= \underbrace{\|\beta v_k + (1-\beta)y_k - x_*\|^2_{\mathbf{E}[Z]^\dagger}}_{I} + \gamma^2 \underbrace{\|Z_k(y_k - x_*)\|^2_{\mathbf{E}[Z]^\dagger}}_{II} \\ &\quad -2\gamma \underbrace{\langle \beta(v_k - x_*) + (1-\beta)(y_k - x_*), \mathbf{E}[Z]^\dagger Z_k(y_k - x_*)\rangle}_{III} \\ &= I + \gamma^2 II - 2\gamma III. \end{aligned} \tag{30}$$

The first term can be upper bounded as follows

$$\begin{aligned} I &= \|\beta(v_k - x_*) + (1-\beta)(y_k - x_*)\|^2_{\mathbf{E}[Z]^\dagger} \\ &= \beta^2\|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + (1-\beta)^2\|y_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + 2\beta(1-\beta)\langle v_k - x_*, y_k - x_*\rangle_{\mathbf{E}[Z]^\dagger} \\ &\overset{(32)}{=} \beta\|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + (1-\beta)\|y_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} - \beta(1-\beta)\|v_k - y_k\|^2_{\mathbf{E}[Z]^\dagger} \\ &\leq \beta r_k^2 + (1-\beta)\|y_k - x_*\|^2_{\mathbf{E}[Z]^\dagger}, \end{aligned} \tag{31}$$

where in the third equality we used a form of the parallelogram identity

$$2\langle u, v\rangle = \|u\|^2 + \|v\|^2 - \|u - v\|^2, \tag{32}$$

with $u = v_k - x_*$ and $v = y_k - x_*$.

Taking expectation with to $\mathcal{S}_k$ in the third term in (30) gives

$$\begin{aligned} \mathbf{E}[III \mid y_k, v_k, x_k] &= \langle \beta v_k + (1-\beta)y_k - x_*, \mathbf{E}[Z]^\dagger \mathbf{E}[Z](y_k - x_*)\rangle \\ &= \langle \beta v_k + (1-\beta)y_k - x_*, y_k - x_*\rangle \qquad (33) \\ &= \langle \beta\left[\frac{1}{\alpha}y_k - \frac{1-\alpha}{\alpha}x_k\right] + (1-\beta)y_k - x_*, y_k - x_*\rangle \\ &= \langle y_k - x_* + \beta\frac{1-\alpha}{\alpha}(y_k - x_k), y_k - x_*\rangle \\ &= \|y_k - x_*\|^2 + \beta\frac{1-\alpha}{\alpha}\langle y_k - x_k, y_k - x_*\rangle \\ &= \|y_k - x_*\|^2 - \beta\frac{1-\alpha}{2\alpha}\left(\|x_k - x_*\|^2 - \|y_k - x_k\|^2 - \|y_k - x_*\|^2\right) \qquad (34) \end{aligned}$$

where in the second equality (33) we used that $y_k - x_* \in \mathbf{Range}(\mathcal{A}^*) \overset{(12)}{=} \mathbf{Range}(\mathbf{E}[Z])$ together with a defining property of pseudoinverse operators $\mathbf{E}[Z]^\dagger \mathbf{E}[Z] w = w$ for all $w \in \mathbf{Range}(\mathbf{E}[Z])$. In the last equality (34) we used yet again the identity (32) with $u = y_k - x_k$ and $v = y_k - x_*$.

Plugging (31) and (34) into (30) and taking conditional expectation gives

$$\begin{aligned} \mathbf{E}\left[r_{k+1}^2 \mid y_k, v_k, x_k\right] &= I + \gamma^2\mathbf{E}[II \mid y_k] - 2\gamma\mathbf{E}[III \mid y_k, v_k, x_k] \\ &\overset{(31)+(34)+(28)}{=} \beta r_k^2 + (1-\beta)\|y_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \gamma^2\nu\|y_k - x_*\|^2_{\mathbf{E}[Z]} \\ &\quad +2\gamma\left(-\|y_k - x_*\|^2 + \beta\frac{1-\alpha}{2\alpha}\left(\|x_k - x_*\|^2 - \|y_k - x_k\|^2 - \|y_k - x_*\|^2\right)\right) \\ &\overset{(29)+(14)}{\leq} \beta r_k^2 + \frac{1-\beta}{\mu}\|y_k - x_*\|^2 + \gamma^2\nu\left(\|y_k - x_*\|^2 - \mathbf{E}\left[\|x_{k+1} - x_*\|^2 \mid y_k\right]\right) \\ &\quad +2\gamma\left(-\|y_k - x_*\|^2 + \beta\frac{1-\alpha}{2\alpha}\left(\|x_k - x_*\|^2 - \|y_k - x_*\|^2\right)\right). \end{aligned} \tag{35}$$

Therefore we have that

$$\mathbf{E}\left[r_{k+1}^2+\gamma^2\nu\|x_{k+1}-x_*\|^2\,|\,y_k,v_k,x_k\right] \;\leq\; \beta\left(r_k^2+\underbrace{\gamma\frac{1-\alpha}{\alpha}}_{P_1}\|x_k-x_*\|^2\right) + \left(\underbrace{\frac{1-\beta}{\mu}-2\gamma+\gamma^2\nu-\beta\gamma\frac{1-\alpha}{\alpha}}_{P_2}\right)\|y_k-x_*\|^2.$$

To establish a recurrence, we need to choose the free parameters γ,α and β so that $P_1=\gamma^2\nu$ and $P_2=0$. Furthermore we should try to set β as small as possible so as to have a fast rate of convergence. Choosing $\beta=1-\sqrt{\frac{\mu}{\nu}}$, $\gamma=\sqrt{\frac{1}{\mu\nu}}$, $\alpha=\frac{1}{1+\gamma\nu}$ gives $P_2=0$, $\gamma^2\nu=1/\mu$ and

$$\mathbf{E}\left[r_{k+1}^2+\tfrac{1}{\mu}\|x_{k+1}-x_*\|^2\,|\,y_k,v_k,x_k\right] \;\leq\; \left(1-\sqrt{\frac{\mu}{\nu}}\right)\left(r_k^2+\tfrac{1}{\mu}\|x_k-x_*\|^2\right). \tag{36}$$

Taking expectation and using the tower rules gives the result. □

B.4 Changing norm

Given an invertible positive self-adjoint $B\in L(\mathcal{X})$, suppose we want to find the least norm solution of (7) under the norm defined by $\|x\|_B\overset{\text{def}}{=}\sqrt{\langle Bx,x\rangle}$ as the metric in $\mathcal{X}$. That is, we want to solve

$$x^*\overset{\text{def}}{=}\arg\min_{x\in\mathcal{X}}\tfrac{1}{2}\|x-x_0\|_B^2,\quad\text{subject to}\quad \mathcal{A}x=b. \tag{37}$$

By changing variables $x=B^{-1/2}z$ we have that the above is equivalent to solving

$$z^*\overset{\text{def}}{=}\arg\min_{z\in\mathcal{X}}\tfrac{1}{2}\|z-z_0\|^2,\quad\text{subject to}\quad \mathcal{A}B^{-1/2}z=b, \tag{38}$$

with $x^*=B^{-1/2}z^*$, and $B^{1/2}$ is the unique symmetric square root of B (see Lemma 18). We can now apply Algorithm 1 to solve (38) where $\mathcal{A}B^{-1/2}$ is the system matrix. Let x_k and v_k be the resulting iterates of applying Algorithm 1. To make explicit this change in the system matrix we define the matrix

$$Z_B\overset{\text{def}}{=}B^{-1/2}\mathcal{A}^*\mathcal{S}_k^*(\mathcal{S}_k\mathcal{A}B^{-1}\mathcal{A}^*\mathcal{S}_k^*)^\dagger\mathcal{S}_k\mathcal{A}B^{-1/2},$$

and the constants

$$\mu_B\overset{\text{def}}{=}\inf_{x\in\mathbf{Range}\left(B^{-1/2}\mathcal{A}^*\right)}\frac{\langle\mathbf{E}\left[Z_B\right]x,x\rangle}{\langle x,x\rangle} \tag{39}$$

and

$$\nu_B\overset{\text{def}}{=}\sup_{x\in\mathbf{Range}\left(B^{-1/2}\mathcal{A}^*\right)}\frac{\langle\mathbf{E}\left[Z_B\mathbf{E}\left[Z_B\right]^\dagger Z_B\right]x,x\rangle}{\langle\mathbf{E}\left[Z_B\right]x,x\rangle}. \tag{40}$$

Theorem 3 then guarantees that

$$\mathbf{E}\left[\|v_{k+1}-z_*\|^2_{\mathbf{E}[Z_B]^\dagger}+\frac{1}{\mu_B}\|x_{k+1}-z_*\|^2\right]\leq\left(1-\sqrt{\frac{\mu_B}{\nu_B}}\right)\mathbf{E}\left[\|v_k-z_*\|^2_{\mathbf{E}[Z_B]^\dagger}+\frac{1}{\mu_B}\|x_k-z_*\|^2\right].$$

Reversing our change of variables $\bar{x}_k=B^{-1/2}x_k$ and $\bar{v}_k=B^{-1/2}v_k$ in the above displayed equation gives

$$\begin{aligned}&\mathbf{E}\left[\|\bar{v}_{k+1}-x_*\|^2_{B^{1/2}\mathbf{E}[Z_B]^\dagger B^{1/2}}+\frac{1}{\mu_B}\|\bar{x}_{k+1}-x_*\|_B^2\right]\\&\quad\leq\left(1-\sqrt{\frac{\mu_B}{\nu_B}}\right)\mathbf{E}\left[\|\bar{v}_k-x_*\|^2_{B^{1/2}\mathbf{E}[Z_B]^\dagger B^{1/2}}+\frac{1}{\mu_B}\|\bar{x}_k-x_*\|_B^2\right].\end{aligned} \tag{41}$$

Thus we recover the same exact from the main theorem in [24], but in a much more general setting.

C Proof of Corollary 4

Clearly, $Z = \frac{1}{A_{i,i}} A^{\frac{1}{2}} S S^\top A^{\frac{1}{2}}$, and hence $\mathbf{E}[Z] = \frac{A}{\mathbf{Tr}(A)}$ and $\mu^P = \frac{\lambda_{\min}(A)}{\mathbf{Tr}(A)}$. After simple algebraic manipulations we get

$$\mathbf{E}\left[\mathbf{E}[Z]^{-\frac{1}{2}} Z \mathbf{E}[Z]^{-1} Z \mathbf{E}[Z]^{-\frac{1}{2}}\right] = \mathbf{Tr}(A)^2 \mathbf{E}\left[\frac{1}{A_{i,i}^2} S S^\top S S^\top\right] = \mathbf{Tr}(A) \mathbf{Diag}\left(A_{i,i}^{-1}\right),$$

and therefore $\nu^P = \lambda_{\max} \mathbf{E}\left[\mathbf{E}[Z]^{-\frac{1}{2}} Z \mathbf{E}[Z]^{-1} Z \mathbf{E}[Z]^{-\frac{1}{2}}\right] = \frac{\mathbf{Tr}(A)}{\min_i A_{i,i}}$.

D Adding a stepsize ω

In this section we enrich Algorithm 1 with several *additional* parameters and study their effect on convergence of the resulting method.

First, we consider an extension of Algorithm 1 to a variant which uses a *stepsize parameter* $0 < \omega < 2$. That is, instead of performing the update

$$x_{k+1} = y_k - g_k, \tag{42}$$

we perform the update

$$x_{k+1} = y_k - \omega g_k. \tag{43}$$

Parameters α, β, γ are adjusted accordingly. The resulting method enjoys the rate $\mathcal{O}\left(\left(1 - \sqrt{\frac{\nu}{\mu}\omega(2-\omega)}\right)^k\right)$, recovering the rate from Theorem 3 as a special case for $\omega = 1$. The formal statement follows.

Theorem 9. *Let $0 < \omega < 2$ be an arbitrary stepsize and define*

$$\eta \overset{def}{=} 2\omega - \omega^2 \geq 0\,. \tag{44}$$

Consider a modification of Algorithm 1 where instead of (42) *we perform the update* (43)*. If we use the parameters*

$$\alpha = \tfrac{1}{1+\gamma\nu} \qquad \beta = 1 - \sqrt{\tfrac{\mu\eta}{\nu}} \qquad \gamma = \sqrt{\tfrac{\eta}{\mu\nu}}, \tag{45}$$

then the iterates $\{v_k, x_k\}_{k\geq 0}$ of Algorithm 1 satisfy

$$\mathbf{E}\left[\|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \tfrac{1}{\mu}\|x_k - x_*\|^2\right] \leq \left(1 - \sqrt{\tfrac{\mu\eta}{\nu}}\right)^k \mathbf{E}\left[\|v_0 - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \tfrac{1}{\mu}\|x_0 - x_*\|^2\right].$$

Proof. See Appendix F. □

E Allowing for different α

In this section we study how the choice of the key parameter α affects the convergence rate.

This parameter determines how much the sequence $y_k = \alpha v_k + (1-\alpha)x_k$ resembles the sequence given by x_k or by v_k. For instance, when $\alpha = 0$, $y_k \equiv x_k$, i.e., we recover the steps of the non-accelerated method, and thus one would expect to obtain the same convergence rate as the non-accelerated method. Similar considerations hold in the other extreme, when $\alpha \to 1$. We investigate this hypothesis, and especially discuss how β and γ must be chosen as a function of α to ensure convergence.

The following statement is a generalization of Theorem 3. For simplicity, we assume that the optional stepsize that was introduced in Theorem 9 is set to one again, $\omega \equiv 1$.

Theorem 10. *Let $0 < \alpha < 1$ be fixed. Then the iterates $\{v_k, x_k\}_{k\geq 0}$ of Algorithm 1 with parameters*

$$\beta(s) = \frac{1 + s - s\sqrt{\frac{\nu + 4\mu s - 2\nu s + \nu s^2}{\nu s^2}}}{2s}, \qquad \gamma(s) = \frac{1}{(1 - s\beta(s))\nu}. \tag{46}$$

where $\tau \overset{def}{=} \frac{1-\alpha}{\alpha}$ *and* $s \overset{def}{=} \frac{\tau}{\beta\gamma}$*, satisfy*

$$\mathbf{E}\left[\|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \gamma\tau\|x_k - x_*\|^2\right] \leq \rho^k \mathbf{E}\left[\|v_0 - x_*\|^2_{\mathbf{E}[Z]^\dagger} + \gamma\tau\|x_0 - x_*\|^2\right].$$

(or put differently):

$$\mathbf{E}\left[\|v_k - x_*\|^2_{\mathbf{E}[Z]^\dagger} + (1-\alpha)\gamma\|x_k - x_*\|^2\right] \leq \rho^k \mathbf{E}\left[\|v_0 - x_*\|^2_{\mathbf{E}[Z]^\dagger} + (1-\alpha)\gamma\|x_0 - x_*\|^2\right].$$

where $\rho = \max\{\beta(s), s\beta(s)\} \leq 1$.

We can now exemplify a few special parameter settings.

Example 11. *For* $\alpha = 1$*, i.e., if* $s \to 0$*, we get the rate* $\rho = 1 - \frac{\mu}{\nu}$ *with* $\beta = 1 - \frac{\mu}{\nu}$*,* $\gamma = \frac{1}{\nu}$*.*

Example 12. *For* $\alpha \to 0$*, i.e., in the limit* $s \to \infty$*, we get the rate* $\rho = 1 - \frac{\mu}{\nu}$*.*

Example 13. *The rate* ρ *is minimized for* $s = 1$*, i.e.,* $\beta = 1 - \sqrt{\frac{\nu}{\mu}}$ *and* $\gamma = \sqrt{\frac{1}{\mu\nu}}$*; recovering Theorem 3.*

The best case, in terms of convergence rate for both non-unit stepsize and a variable parameter choice happened to be the default parameter setup. The non-optimal parameter choice was studied in order to have theoretical guarantees for a wider class of parameters, as in practice one might be forced to rely on sub-optimal / inexact parameter choices.

F Proof of Theorem 9

The proof follows by slight modifications of the proof of Theorem 3.

First we adapt Lemma 8. As we have $x_{k+1} - x_* = (1 - \omega Z_k)(y_k - x_*)$ the following statement follows by the same arguments as in the proof of Lemma 8.

Lemma 14 (Lemma 8')**.**

$$\eta\|y_k - x_*\|^2_{\mathbf{E}[Z]} = \|y_k - x_*\|^2 - \mathbf{E}\left[\|x_{k+1} - x_*\|^2 \,|\, y_k\right] \tag{47}$$

Proof.

$$\begin{aligned}
\mathbf{E}\left[\|x_{k+1} - x_*\|^2 \,|\, y_k\right] &= \mathbf{E}\left[\|(I - Z_k)(y_k - x_*)\|^2 \,|\, y_k\right] \\
&= \mathbf{E}\left[\langle (I - \omega Z_k)(y_k - x_*), (I - \omega Z_k)y_k - x_*\rangle\right] \\
&= \|y_k - x_*\|^2 - \eta\|y_k - x_*\|^2_{\mathbf{E}[Z]}.
\end{aligned}$$

□

We now follow the same steps as in proof of Theorem 3 in Section B.3. We observe, that the first time Lemma 8 is applied is in equation (35). Using Lemma 14 instead, gives

$$\begin{aligned}
\mathbf{E}\left[r^2_{k+1} \,|\, y_k, v_k, x_k\right] \;\leq\; & \beta r_k^2 + \frac{1-\beta}{\mu}\|y_k - x_*\|^2 + \frac{\gamma^2\nu}{\eta}\left(\|y_k - x_*\|^2 - \mathbf{E}\left[\|x_{k+1} - x_*\|^2 \,|\, y_k\right]\right) \\
& + 2\gamma\left(-\|y_k - x_*\|^2 + \beta\frac{1-\alpha}{2\alpha}\left(\|x_k - x_*\|^2 - \|y_k - x_*\|^2\right)\right).
\end{aligned} \tag{48}$$

Therefore we have that

$$\begin{aligned}
\mathbf{E}\left[r^2_{k+1} + \gamma^2\nu\|x_{k+1} - x_*\|^2 \,|\, y_k, v_k, x_k\right] \;\leq\; & \beta\left(r_k^2 + \underbrace{\gamma\frac{1-\alpha}{\alpha}}_{P_1'}\|x_k - x_*\|^2\right) \\
& + \left(\underbrace{\frac{1-\beta}{\mu} - 2\gamma + \frac{\gamma^2\nu}{\eta} - \beta\gamma\frac{1-\alpha}{\alpha}}_{P_2'}\right)\|y_k - x_*\|^2.
\end{aligned}$$

Noting that $\frac{1-\alpha}{\alpha} = \gamma\nu$ and $\frac{\gamma^2\nu}{\eta} = \frac{\gamma(1-\alpha)}{\eta\alpha} = \frac{1}{\mu}$, we observe $P_2' = 0$ and deduce the statement of Theorem 9.

G Proof of Theorem 10

It suffices to study equation (35). We observe that for convergence the big bracket, P_2, should be negative,

$$(1-\beta)\frac{1}{\mu}+\gamma^2\nu-2\gamma-\gamma\beta\frac{1-\alpha}{\alpha}\leq 0 \tag{49}$$

The convergence rate is then

$$\rho \overset{\text{def}}{=} \max\left\{\beta, \frac{(1-\alpha)\beta}{\alpha\gamma\nu}\right\}. \tag{50}$$

or in the notation of Theorem 10, $\rho=\max\{\beta, s\beta\}$.

This means, that in order to obtain the best convergence rate, we should therefore choose parameters β and γ such that β is as small as possible. This observation is true regardless of the value of s (which itself depends on γ).

With the notation $\tau=s\gamma\beta$, we reformulate (49) to obtain

$$\frac{1}{\mu}+\gamma^2\nu-2\gamma\leq\beta\left(\frac{1}{\mu}+s\gamma^2\nu\right) \tag{51}$$

Thus we see, that β cannot be chosen smaller than

$$\beta^\star(s,\gamma)=\frac{1+\mu\gamma^2\nu-2\mu\gamma}{1+s\mu\gamma^2\nu} \tag{52}$$

Minimizing this expression in γ gives

$$\beta^\star(s)=\frac{1+s-s\sqrt{\frac{\nu+4\mu s-2\nu s+\nu s^2}{\nu s^2}}}{2s} \tag{53}$$

with $\gamma^\star(s)=\frac{1}{(1-s\beta^\star(s))\nu}$.

We further observe that this parameter setting indeed guarantees convergence, i.e. $\rho\leq 1$. From (53) we observe ($\nu>0$, $s\geq 0$, $\mu\geq 0$):

$$\beta^\star(s)\leq\frac{1+s-\sqrt{\frac{\nu-2\nu s+\nu s^2}{\nu}}}{2s}=\frac{1+s-(s-1)}{2s}=\frac{1}{s} \tag{54}$$

Hence $s\beta^\star(s)\leq 1$. On the other hand, $(1-s)\leq\sqrt{(1-s)^2+\frac{4\mu s}{\nu}}$ and hence $(1+s)-\sqrt{(1-s)^2+\frac{4\mu s}{\nu}}\leq 2s$, which shows $\beta^\star(s)\leq 1$.

H Proofs and Further Comments on Section 4

H.1 Proof of Theorem 5

We perform a change of coordinates since it is easier to work with the standard Frobenius norm as opposed to the weighted Frobenius norm. Let $\hat{X}=A^{1/2}XA^{1/2}$ so that (18) and (20) become

$$\hat{X}_* \overset{\text{def}}{=} I=\arg\min\|\hat{X}\|_F^2 \quad \text{subject to} \quad \hat{X}=I, \quad \hat{X}=\hat{X}^\top, \tag{55}$$

and

$$\hat{X}_{k+1}=P+(I-P)\,\hat{X}_k\,(I-P), \tag{56}$$

respectively, where $P=A^{1/2}S(S^\top AS)^{-1}S^\top A^{1/2}$. The linear operator that encodes the constaint in (4.2) is given by $\hat{\mathcal{A}}(X)=\left(X,\, X-X^\top\right)$ the adjoint of which is given by $\hat{\mathcal{A}}^*(Y_1,Y_2)=Y_1+Y_2-Y_2^\top$. Since $\hat{\mathcal{A}}^*$ is clearly surjective, it follows that $\mathbf{Range}\left(\hat{\mathcal{A}}^*\right)=\mathbb{R}^{n\times n}$.

Subtracting the identity matrix from both sides of (56) and using that P is a projection matrix, we have that

$$\hat{X}_{k+1} - I = (I - P)\left(\hat{X}_k - I\right)(I - P). \tag{57}$$

To determine the Z operator (9), from (11) and (57) we know that

$$(I - P)\left(\hat{X}_k - I\right)(I - P) = (I - Z)(\hat{X}_k - I).$$

Thus for every matrix $X \in \mathbb{R}^{n\times n}$ we have that

$$Z(X) = X - (I - P)\,X\,(I - P) = XP + PX(I - P). \tag{58}$$

Denote column-wise vectorization of X as x: $x \overset{\text{def}}{=} \mathbf{Vec}\,(X)$. To calculate a useful lower bound on μ, note that

$$\begin{aligned}
\mathbf{Tr}\left(X^\top Z(X)\right) &= \mathbf{Tr}\left(X^\top XP\right) + \mathbf{Tr}\left(X^\top PX(I-P)\right) \\
&= x^\top \mathbf{Vec}\,(XP) + x^\top \mathbf{Vec}\,(PX(I-P)) \\
&= x^\top (P\otimes I)x + x^\top ((I-P)\otimes P)x \\
&\overset{(23)}{=} x^\top \mathbf{Z} x,
\end{aligned} \tag{59}$$

where we used that $\mathbf{Tr}\left(A^\top B\right) = \mathbf{Vec}\,(A)^\top\,\mathbf{Vec}\,(B)$ and $\mathbf{Vec}\,(AXB) = (B^\top \otimes A)\mathbf{Vec}\,(x)$ holds for any A, B, X.

Consequently, μ is equal to

$$\mu \overset{(??)}{=} \inf_{X\in\mathbb{R}^{n\times n}} \frac{\langle \mathbf{E}\left[Z\right]X, X\rangle_F}{\|X\|_F^2} \overset{(59)}{=} \inf_{x\in\mathbb{R}^{n^2\times n^2}} \frac{x^\top \mathbf{E}\left[\mathbf{Z}\right]x}{x^\top x} = \lambda_{\min}(\mathbf{E}\left[\mathbf{Z}\right]).$$

Notice that we have $2\lambda_{\min}(\mathbf{E}\left[P\right]) \geq \lambda_{\min}(\mathbf{E}\left[\mathbf{Z}\right]) \geq \lambda_{\min}(\mathbf{E}\left[P\right])$ since $(P\otimes I) + (I\otimes P) \geq \mathbf{Z} \geq (P\otimes I)$.

In light of Algorithm 1, the iterates of the accelerated version of (56) are given by

$$\begin{aligned}
\hat{Y}_k &= \alpha\hat{V}_k + (1-\alpha)\hat{X}_k \\
\hat{G}_k &= Z_k(\hat{Y}_k - I) \\
\hat{X}_{k+1} &= \hat{Y}_k - \hat{G}_k \\
\hat{V}_{k+1} &= \beta\hat{V}_k + (1-\beta)\hat{Y}_k - \gamma\hat{G}_k
\end{aligned} \tag{60}$$

where $\hat{Y}_k, \hat{V}_k, \hat{G} \in \mathbb{R}^{n\times n}$. From Theorem 3 we have that $\hat{V}_k$ and $\hat{X}_k$ converge to the identity matrix according to

$$\mathbf{E}\left[\|\hat{V}_{k+1} - I\|^2_{\mathbf{E}[Z]^\dagger} + \frac{1}{\mu}\|\hat{X}_{k+1} - I\|_F^2\right] \leq \left(1 - \sqrt{\frac{\mu}{\nu}}\right)\mathbf{E}\left[\|\hat{V}_k - I\|^2_{\mathbf{E}[Z]^\dagger} + \frac{1}{\mu}\|\hat{X}_k - I\|_F^2\right], \tag{61}$$

where $\|X\|^2_{\mathbf{E}[Z]^\dagger} = \langle \mathbf{E}\left[Z\right]^\dagger X, X\rangle_F$. Changing coordinates back to $\hat{X}_k = A^{1/2}X_kA^{1/2}$ and defining $Y_k \overset{\text{def}}{=} A^{-1/2}\hat{Y}_kA^{-1/2}$, $V_k \overset{\text{def}}{=} A^{-1/2}\hat{V}_kA^{-1/2}$ and $G_k \overset{\text{def}}{=} A^{-1/2}\hat{G}_kA^{-1/2}$, we have that (61) gives (21). Furthermore, using the same coordinate change applied to the iterates (60) gives Algorithm 2.

H.2 Matrix inversion as linear system

Denote $x = \mathbf{Vec}\,(X)$, i.e. x is n^2 dimensional vector such that $X_{(n(i-1)+1):ni} = X_{:,i}$. Similarly, denote $e = \mathbf{Vec}\,(I)$. System (6) can be thus rewritten as

$$(I\otimes A)x = e. \tag{62}$$

Notice that all linear sketches of the original system $AX = I$ can be written as

$$S_0{}^\top(I\otimes A)x = S_0{}^\top e \tag{63}$$

for a suitable $n^2\times n^2$ matrix S_0, therefore the setting is fairly general.

H.2.1 Alternative proof of Theorem 5

Let us now, for a purpose of this proof, consider sketch matrix S_0 to capture only sketching the original matrix system $AX = I$ by left multiplying by S, i.e. $S_0 = (I \otimes S)$, as those are the considered sketches in the setting of Section 4.

As we have

$$\mathbf{Tr}\left(BX^\top BX\right) = \mathbf{Vec}\left(BXB\right)^\top x = x^\top (B \otimes B)x,$$

weighted Frobenius norm of matrices is equivalent to a special weighted euclidean norm of vectors. Define also C to be a matrix such that $Cx = 0$ if and only if $X = X^\top$. Therefore, (4.2) is equivalent to

$$x_{k+1} = \arg\min \|x - x_k\|^2_{A\otimes A} \quad \text{subject to} \quad (I \otimes S^\top)(I \otimes A)x = (I \otimes S^\top)e, \quad Cx = 0, \quad (64)$$

which is a sketch-and-project method applied on the linear system, with update as per (20):

$$x^{k+1} = x^k - (H \otimes I)((I \otimes A)x - e) - (I \otimes H)((I \otimes A)x - e) + (HA \otimes H)((I \otimes A)x - e)$$

for $H \overset{\text{def}}{=} S\left(S^\top AS\right)^{-1} S^\top$. Using substitution $\hat{x} = (A^{\frac{1}{2}} \otimes A^{\frac{1}{2}})x$; $\hat{S} = A^{\frac{1}{2}}S$ and comparing to (11), we get

$$Z = I \otimes I - (I - P) \otimes (I - P)$$

for P as defined inside the statement of Theorem 5. Therefore, we have all necessary information to apply the results from [24], recovering Theorem 5.

I Linear Operators in Euclidean Spaces

Here we provide some technical lemmas and results for linear operators in Euclidean space, that we used in the main body of the paper. Most of these results can be found in standard textbooks of analysis, such as [22]. We give them here for completion.

Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be Euclidean spaces, equipped with inner products. Formally, we should use a notation that distinguishes the inner product in each space. But instead we use $\langle \cdot, \cdot \rangle$ to denote the inner product on all spaces, as it will be easy to determine from which space the elements are in. That is, for $x_1, x_2 \in \mathcal{X}$, we denote by $\langle x_1, x_2 \rangle$ the inner product between x_1 and x_2 in $\mathcal{X}$.

Let

$$\|T\| \overset{\text{def}}{=} \sup_{\|x\| \leq 1} \|Tx\|,$$

denote the operator norm of T. Let $0 \in L(\mathcal{X}, \mathcal{Y})$ denote the zero operator and $I \in L(\mathcal{X}, \mathcal{Y})$ the identity map.

The adjoint. Let $T^* \in L(\mathcal{Y}, \mathcal{X})$ denote the unique operator that satisfies

$$\langle Tx, y \rangle = \langle x, T^* y \rangle,$$

for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. We say that T^* is the *adjoint* of T. We say T is *self-adjoint* if $T = T^*$. Since for all $x \in \mathcal{X}$ and $s \in \mathcal{S}$,

$$\langle x, (ST)^* s \rangle = \langle STx, s \rangle_{\mathcal{S}} = \langle Tx, S^* s \rangle_{\mathcal{Y}} = \langle x, T^* S^* s \rangle,$$

we have

$$(ST)^* = T^* S^*.$$

Lemma 15. *For $T \in L(\mathcal{X}, \mathcal{Y})$ we have that $\mathbf{Range}\left(T^*\right)^\perp = \mathbf{Null}\left(T\right)$. Thus*

$$\mathcal{X} = \mathbf{Range}\left(T^*\right) \oplus \mathbf{Null}\left(T\right) \quad (65)$$

$$\mathcal{Y} = \mathbf{Range}\left(T\right) \oplus \mathbf{Null}\left(T^*\right) \quad (66)$$

Proof. See 3.2.6 in [22]. □

I.1 Positive Operators

We say that $G \in L(\mathcal{X})$ is positive if it is self-adjoint and if $\langle x, Gx\rangle \geq 0$ for all $x \in \mathcal{X}$. Let $(e_j)_{j=1}^{\infty} \in \mathcal{X}$ be an orthonormal basis. The trace of G is defined as

$$\mathbf{Tr}\,(G) \stackrel{\text{def}}{=} \sum_{j=1}^{\infty} \langle Ge_j, e_j\rangle. \tag{67}$$

The definition of trace is independent of the choice of basis due to the following lemma.

Lemma 16. *If U is unitary and $G \geq 0$ then $\mathbf{Tr}\,(UGU^*) = \mathbf{Tr}\,(G)$.*

Proof. See 3.4.3 and 3.4.4 in [22]. □

Lemma 17. *If $P \in L(\mathcal{X})$ is a projection matrix then $\mathbf{Tr}\,(P) = \dim(\mathbf{Range}\,(P)) = \mathbf{Rank}\,(P)$.*

Proof. Let $d = \dim(\mathbf{Range}\,(P))$ which is possibly infinite. Given that P is a projection we have that $\mathbf{Range}\,(P)$ is a closed subspace and thus there exists orthonormal basis $(e_j)_{j=1}^{d}$ of $\mathbf{Range}\,(P)$. Consequently, $\mathbf{Tr}\,(P) \stackrel{(67)}{=} \sum_{j=1}^{d} 1 = d = \dim(\mathbf{Range}\,(P))$. □

A *square root* of an operator $G \in L(\mathcal{X})$ is an operator $R \in L(\mathcal{X})$ such that $R^2 = G$.

Lemma 18. *If $G : \mathcal{X} \to \mathcal{X}$ is positive, then there exists a unique positive square root of G which we denote by $G^{1/2}$.*

Proof. See 3.2.11 in [22]. □

Lemma 19. *For any $T \in L(\mathcal{X}, \mathcal{Y})$ and any $G \in L(\mathcal{Y}, \mathcal{Y})$ that is positive and injective,*

$$\mathbf{Null}\,(T) = \mathbf{Null}\,(T^*GT)\,, \tag{68}$$

and

$$\overline{\mathbf{Range}\,(T^*)} = \overline{\mathbf{Range}\,(T^*GT)}. \tag{69}$$

Proof. The inclusion $\mathbf{Null}\,(T) \subset \mathbf{Null}\,(T^*GT)$ is immediate. For the opposite inclusion, let $x \in \mathbf{Null}\,(T^*GT)$. Since G is positive we have by Lemma 18 that there exists a square root with $G^{1/2}G^{1/2} = G$. Therefore, $\langle x, T^*GTx\rangle = \langle G^{1/2}Tx, G^{1/2}Tx\rangle = 0$, which implies that $G^{1/2}Tx = 0$. Since G is injective, it follows that $G^{1/2}$ is injective and thus $x \in \mathbf{Null}\,(T)$. Finally (69) follows by taking the orthogonal complements of (68) and observing Lemma 15. □

As an immediate consequence of (68) and (69) we have the following lemma.

Corollary 20. *For $G : \mathcal{X} \to \mathcal{X}$ positive we have that*

$$\mathbf{Null}\left(G^{1/2}\right) = \mathbf{Null}\,(G) \tag{70}$$

$$\overline{\mathbf{Range}\left(G^{1/2}\right)} = \overline{\mathbf{Range}\,(G)} \tag{71}$$

I.2 Pseudoinverse

For a bounded linear operator T define the pseudoinverse of T as follows.

Definition 21. *Let $T \in L(\mathcal{X}, \mathcal{Y})$ such that $\mathbf{Range}\,(T)$ is closed. $T^\dagger : \mathcal{Y} \to \mathcal{X}$ is said to be the pseudoinverse if*

i) $T^\dagger Tx = x$ for all $x \in \mathbf{Range}\,(T^*)$.

ii) $T^\dagger x = 0$ for all $x \in \mathbf{Null}\,(T^*)$.

iii) If $x \in \mathbf{Null}\,(T)$ and $y \in \mathbf{Range}\,(T^*)$ then $T^\dagger(x + y) = T^\dagger x + T^\dagger y$.

It follows directly from the definition (see [6] for details) that $T^\dagger$ is a unique bounded linear operator. The following properties of pseudoinverse will be important.

Lemma 22 (Properties of pseudoinverse)**.** *Let $T \in L(\mathcal{X}, \mathcal{Y})$ such that $\mathbf{Range}(T)$ is closed. It follows that*

i) $TT^\dagger T = T$

ii) $\mathbf{Range}\left(T^\dagger\right) = \mathbf{Range}\left(T^*\right)$ *and* $\mathbf{Null}\left(T^\dagger\right) = \mathbf{Null}\left(T^*\right)$

iii) $(T^*)^\dagger = (T^\dagger)^*$

iv) If T is self-adjoint and positive then $T^\dagger$ is self-adjoint and positive.

v) $T^\dagger T T^* = T^*$*, that is, $T^\dagger T$ projects orthogonally onto $\mathbf{Range}\left(T^*\right)$ and along $\mathbf{Null}\left(T\right)$.*

vi) Consider the linear system $Tx = d$ where $d \in \mathbf{Range}(T)$. It follows that

$$T^\dagger d = \arg\min_{x \in \mathcal{X}} \tfrac{1}{2}\|x\|^2 \quad \textit{subject to} \quad Tx = d. \tag{72}$$

vii) $T^\dagger = T^*(TT^*)^\dagger$

Proof. The proof of items *i, ii, iii, iv, v* can be found in [6]. The proof of item *vi* is alternative characterization of the pseudoinverse and it can be established by using that $d \in \mathbf{Range}(T)$ together with item *i* thus $TT^\dagger d = d$. The proof then follows by using the orthogonal decomposition $\mathbf{Range}\left(T^*\right) \oplus \mathbf{Null}\left(T\right)$ to show that $T^\dagger d$ is indeed the minimum of (72). Finally item (vii) is a direct consequence of the previous items. □

J Conclusions and Extensions

We developed an accelerated sketch-and-project method for solving linear systems in Euclidean spaces. The method was applied to invert positive definite matrices, while keeping their symmetric structure. Our accelerated matrix inversion algorithm was then incorporated into an optimization framework to develop both accelerated stochastic and deterministic BFGS, which to the best of our knowledge, are *the first accelerated quasi-Newton updates.*

We show that under a careful choice of the parameters of the method, and depending on the problem structure and conditioning, acceleration might result into significant speedups both for the matrix inversion problem and for the stochastic BFGS algorithm. We confirm experimentally that our accelerated methods can lead to speed-ups when compared to the classical BFGS algorithm.

As a future line of research, it might be interesting to study the accelerated BFGS algorithm (either deterministic or stochastic) further, and provide a convergence analysis on a suitable class of functions. Another interesting area of research might be to combine accelerated BFGS with limited memory [14] or engineer the method so that it can efficiently compete with first order algorithms for some empirical risk minimization problems, such as, for example [9].

As we show in this work, *Nesterov's acceleration can be applied to quasi-Newton updates*. We believe this is a surprising fact, as quasi-Newton updates have not been understood as optimization algorithms, which prevented the idea of applying acceleration in this context.

Since since second-order methods are becoming more and more ubiquitous in machine learning and data science, we hope that our work will motivate further advances at the frontiers of big data optimization.