Projected Stochastic Momentum Methods for Nonlinear Equality-Constrained Optimization for Machine Learning

Qi Wang

Department of Industrial and Operations Engineering, University of Michigan

Christian Piermarini

Amazon Sàrl

Yunlang Zhu, Frank E. Curtis

Department of Industrial and Systems Engineering, Lehigh University

COR@L Technical Report 26T-001

Projected Stochastic Momentum Methods for Nonlinear Equality-Constrained Optimization for Machine Learning

Qi Wang[1], Christian Piermarini[2], and Yunlang Zhu, Frank E. Curtis[3]

[1]Department of Industrial and Operations Engineering, University of Michigan*
[2]Amazon Sàrl†
[3]Department of Industrial and Systems Engineering, Lehigh University‡§

Jan 16, 2026

Abstract

Two algorithms are proposed, analyzed, and tested for solving continuous optimization problems with nonlinear equality constraints. Each is an extension of a stochastic momentum-based method from the unconstrained setting to the setting of a stochastic Newton-SQP-type algorithm for solving equality-constrained problems. One is an extension of the heavy-ball method and the other is an extension of the Adam optimization method. Convergence guarantees for the algorithms for the constrained setting are provided that are on par with state-of-the-art guarantees for their unconstrained counterparts. A critical feature of each extension is that the momentum terms are implemented with projected gradient estimates, rather than with the gradient estimates themselves. The significant practical effect of this choice is seen in an extensive set of numerical experiments on solving informed supervised machine learning problems. These experiments also show benefits of employing a constrained approach to supervised machine learning rather than a typical regularization-based approach.

1 Introduction

Algorithms based on the stochastic-gradient methodology [27, 28] have been found to be especially powerful for solving modern-day unconstrained continuous optimization problems that arise in multiple areas, most notably in supervised machine learning. Of the many variants of the stochastic-gradient methodology, momentum-based approaches have been particularly popular and effective in practice. These include the heavy-ball method [25], Adagrad [12], RMSprop [9, 30], and Adam [15]; see [3] for an overview.

The main contributions of this paper are extensions of the heavy-ball and Adam approaches from the unconstrained setting to the setting of a stochastic Newton-SQP framework for solving nonlinear-equality-constrained continuous optimization problems. Our approaches are inspired by the stochastic Newton-SQP method proposed in [2] (see also [1, 8]) for solving problems with nonlinear equality constraints. We show that our proposed methods can offer theoretical guarantees that are on par with state-of-the-art guarantees that have been offered for the heavy-ball method and Adam in the unconstrained setting. In particular, our analyses follow the analyses for these methods in the unconstrained setting that are presented in [10].

*qiwangqi@umich.edu
†chripiermarini@gmail.com
‡yuza23@lehigh.edu
§frank.e.curtis@lehigh.edu

Due to its more impressive practical performance, our more significant contribution in this paper is the extension of Adam to the equality-constrained setting. A different extension of Adam to the equality-constrained setting (without a convergence guarantee) has been proposed previously; see [21]. However, besides the fact that we offer a theoretical convergence guarantee for it, our algorithm is unique in that the running averages that are maintained in an Adam-based approach are taken with *projected* gradient estimates rather than with gradient estimates themselves. This is also the case with our extension of the heavy-ball method. We show in an extensive set of numerical experiments with informed supervised machine learning test problems that our projected stochastic Adam algorithm can outperform both (a) Adam applied to minimize a regularized objective function and (b) the projection-less extension of Adam proposed in [21].

A variant of the stochastic Newton-SQP method from [2] that employs Adagrad-type scaling has been proposed and analyzed in [24]. However, the algorithm and analysis in this paper are distinct from those in [24] due to the subtle, yet significant differences between convergence analyses for Adagrad- versus other methods. As far as we are aware, ours is the first paper that offers convergence guarantees for heavy-ball- and Adam-based stochastic Newton-SQP methods for solving nonlinear-equality-constrained problems.

1.1 Outline

The class of optimization problems of our interest and fundamental properties of the algorithms that we propose and analyze are presented in §2. Our proposed heavy-ball- and Adam-based schemes, along with our convergence analyses of them, are given in §3. In §4, we discuss a broad class of problems, namely, informed supervised machine learning problems, for which the proposed algorithms are particularly well suited. In that section we also discuss a few critical considerations for efficient implementations of the algorithms. In §5, we present the results of a large set of experiments that demonstrate the effectiveness of the algorithms. Finally, we provide some concluding remarks in §6.

2 Problem Formulation and Stochastic Newton-SQP Framework

Our problem class of interest is that of continuous equality-constrained optimization problems, where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ and constraint function $c : \mathbb{R}^n \to \mathbb{R}^m$ are continuously differentiable. Our proposed algorithms are designed to solve such problems when the objective function is defined by an expectation of a function with a random variable argument and may be nonconvex, and when the equality-constraint function may be nonlinear. Formally, our problem class of interest can be expressed as instances of

$$\min_{x\in\mathbb{R}^n} \; f(x) \;\; \text{subject to} \;\; c(x) = 0, \;\; \text{where} \;\; f(x) = \mathbb{E}_\xi[F(x,\xi)] \;\; \text{for all} \;\; x \in \mathbb{R}^n, \tag{1}$$

ξ is a random variable with associated probability space $(\Xi, \mathcal{F}_\xi, \mathbb{P}_\xi)$, $F : \mathbb{R}^n \times \Xi \to \mathbb{R}$, and $\mathbb{E}_\xi$ denotes the expected value operator with respect to the probability measure $\mathbb{P}_\xi$. At a given $x \in \mathbb{R}^n$, a first-order optimality condition for problem (1) is that there exists a Lagrange multiplier $y \in \mathbb{R}^m$ such that

$$\nabla f(x) + J(x)^T y = 0 \;\; \text{and} \;\; c(x) = 0, \;\; \text{where} \;\; J := \nabla c^T. \tag{2}$$

The first of these conditions states that the gradient $\nabla f(x)$ lies in the range space of the constraint derivative $J(x)^T$. Due to the fundamental theorem of linear algebra, this is equivalent to the property that the projection of the gradient $\nabla f(x)$ onto the null space of the constraint Jacobian $J(x)$ is equal to the zero vector. Under the assumption that the constraint Jacobian has full row rank and with $P(x)$ denoting the projection operator onto this null space at x, the first-order optimality conditions in (2) are equivalent to

$$P(x)\nabla f(x) = 0 \;\; \text{and} \;\; c(x) = 0, \;\; \text{where} \;\; P(x) = I - J(x)^T(J(x)J(x)^T)^{-1}J(x). \tag{3}$$

These are the form of the first-order optimality conditions that we employ in our analyses.

The algorithms that we propose, analyze, and test in this paper have as a basis the stochastic-gradient-based Sequential Quadratic Programming (SQP) framework proposed and analyzed in [2] for solving equality-constrained optimization problems. A simplified version of this method is stated as Algorithm 1. The key

aspect of it that is distinct for the setting of constrained optimization is that each search direction is computed to satisfy a linearization of the constraints that is defined with respect to the current iterate x_k. Specifically, given an objective gradient estimate $g_k \in \mathbb{R}^n$ and a symmetric and positive-definite matrix $H_k \in \mathbb{R}^{n\times n}$, the search direction d_k can equivalently be defined as the solution of the quadratic optimization subproblem

$$\min_{d\in\mathbb{R}^n} \; g_k^T d + \tfrac{1}{2} d^T H_k d \;\text{ s.t. }\; c(x_k) + J(x_k)d = 0. \tag{4}$$

The linear system (5) represents the necessary and sufficient conditions for optimality for (4), so any solution of (5) yields a solution of (4) as well as a Lagrange multiplier estimate y_k. (If $x \equiv x_k$ is a point at which there exists y such that (2) holds and $g_k = \nabla f(x_k)$, then a solution of (5) is $(d_k, y_k) = (0, y)$.) It is well known that if $J(x_k)$ has full row rank and since H_k is positive definite in the null space of $J(x_k)$, subproblem (4) is feasible and has a unique globally optimal solution, which is given by the unique solution of (5).

Algorithm 1 Stochastic-Gradient-based SQP Framework [2]

Require: $x_1 \in \mathbb{R}^n$ and $\{\alpha_k\} \subset (0, 1]$

1: **for** all $k \in \mathbb{N}$ **do**

2: compute a stochastic gradient estimate $g_k \approx \nabla f(x_k)$ and choose symmetric $H_k \in \mathbb{R}^{n\times n}$

3: compute d_k by solving

$$\begin{bmatrix} H_k & J(x_k)^T \\ J(x_k) & 0 \end{bmatrix} \begin{bmatrix} d_k \\ y_k \end{bmatrix} = - \begin{bmatrix} g_k \\ c(x_k) \end{bmatrix} \tag{5}$$

4: set $x_{k+1} \leftarrow x_k + \alpha_k d_k$

5: **end for**

An algorithm that computes each search direction by solving a system of the form (5) can employ a direct solver for symmetric indefinite linear systems. Alternatively, the search direction can be computed using a step decomposition method. This is the approach that we specify for the methods that we propose in this paper since the methods make use of running averages of projected gradient estimates. We close this section with a description of the linear algebra involved in a step decomposition approach, which also reveals the computational cost required to compute search directions in our proposed methods. Importantly, these costs will be small when the number of equality constraints m is small, meaning that in such cases the computational cost of each iteration will be proportional to the costs in an unconstrained context.

Suppose that, similarly as in Algorithm 1, an algorithm computes each search direction through solving

$$\begin{bmatrix} I & J^T \\ J & 0 \end{bmatrix} \begin{bmatrix} s \\ y \end{bmatrix} = - \begin{bmatrix} q \\ c \end{bmatrix}, \tag{6}$$

where I is the identity matrix and we assume that $J \in \mathbb{R}^{m\times n}$ has full row rank. By the fundamental theorem of linear algebra, the solution component s can be expressed as $s = v + u$, where $v \in \text{Range}(J^T)$ and $u \in \text{Null}(J)$. Considering the second block of equations, one finds that v can be computed by solving an m-dimensional positive-definite system $JJ^T\tilde{v} = -c$ for $\tilde{v} \in \mathbb{R}^m$, then computing $J^T\tilde{v} = v$, which yields

$$v = -J^T(JJ^T)^{-1}c. \tag{7}$$

The computational cost of computing v is thus $\mathcal{O}(m^3 + m^2 n)$.

Now letting $Z \in \mathbb{R}^{n\times(n-m)}$ denote an orthogonal matrix whose columns span Null(J), the first row of (6) states $u + J^T y = -(q + v)$, so $u = -Z(Z^TZ)^{-1}Z^Tq$. However, it is not efficient to compute u in this manner since it requires computing the null-space basis matrix Z. Fortunately, one can replace $Z(Z^TZ)^{-1}Z^T$ with a matrix expressed in terms of J. Specifically, one finds that $Z(Z^TZ)^{-1}Z^T = I - J^T(JJ^T)^{-1}J =: P$, so

$$u = -(I - J^T(JJ^T)^{-1}J)q = -Pq. \tag{8}$$

In other words, u is the negative of the projection of q onto the null space of J; recall (3). Practically speaking, the component u can be computed by computing the matrix-vector product $\hat{q} := Jq$, solving the

m-dimensional positive definite system $JJ^T\bar{q} = \hat{q}$ for $\bar{q} \in \mathbb{R}^m$, computing the matrix-vector product $J^T\bar{q}$, and adding the result to $-q$. Thus, similar to v, the cost of computing u is $\mathcal{O}(m^3 + m^2 n)$.

Through this discussion, one can observe that the solution of (6) is the same if q is replaced by Pq. This follows since (7) reveals that q has no effect on v, and since by virtue of P being an orthogonal projection matrix one has by (8) that $u = -Pq = -P^2 q$. That being said, the solution of the system is clearly affected if q is replaced by another vector not necessarily in the null space of the constraint Jacobian. These comments reveal a critical distinction between our proposed version of Adam and that proposed in [21]. In [21], running averages (i.e., the momentum terms) are taken with gradient vectors, whereas in our proposed approach these running averages are taken with projected gradients. This difference is critical theoretically and practically.

3 Stochastic Momentum-based Algorithms

In this section, we propose and analyze two new stochastic momentum-based methods for solving (1). Each algorithm computes search direction components through the formulas (7) and (8). However, they are each distinct in the manner that scaling and momentum are applied to the latter component when constructing the search direction taken by the algorithm.

The stochastic nature of the algorithms means that our analysis of each of them considers a stochastic process defined by each algorithm. In each case, let $(\Omega, \mathcal{F}, \mathbb{P})$ denote a probability space that captures the behavior of the algorithm, which is to say that each outcome in Ω represents a possible realization of a run of the algorithm. In addition, let $\mathbb{E}$ denote the expected value operator defined by the probability measure $\mathbb{P}$. The only source of randomness in each iteration is the computation of a stochastic gradient estimate. Hence, using the notation of (1), one can consider the set of outcomes as $\Omega \equiv \Xi \times \Xi \times \cdots$. Let $\mathcal{F}_1$ be the σ-algebra defined by the initial conditions of an algorithm, and, more generally, for all $k \in \mathbb{N}$ let $\mathcal{F}_k$ be the σ-algebra generated by the initial conditions and the stochastic gradient estimators up through the end of iteration $k-1$. In this manner, one has that $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \cdots \subseteq \mathcal{F}$ and the sequence $\{\mathcal{F}_k\}$ is a filtration.

Throughout this section, we employ the shorthand notation $c_k := c(x_k)$, $J_k := \nabla c(x_k)^T$, and $P_k := P(x_k)$.

3.1 Projected Stochastic Heavy-Ball SQP

We first consider an extension of the heavy-ball method to the setting of equality-constrained optimization. Specifically, we propose Algorithm 2.

Algorithm 2 Projected Stochastic Heavy-ball SQP

Require: $x_1 \in \mathbb{R}^n$, $\{\rho_k\}$ and $\{h_k\}$ with $\rho_k \in (0,1]$ and $h_k \in \mathbb{R}_{>0}$ for all $k \in \mathbb{N}$, $\beta \in [0,1)$, and $\alpha \in (0,1]$

set $r_0 \leftarrow 0 \in \mathbb{R}^n$
for all $k \in \mathbb{N}$ **do**
  compute a stochastic gradient estimate $g_k \approx \nabla f(x_k)$
  compute $v_k \leftarrow -\rho_k J_k^T \left(J_k J_k^T\right)^{-1} c_k$
  compute $u_k \leftarrow -h_k^{-1} P_k g_k$, where $P_k := I - J_k^T (J_k J_k^T)^{-1} J_k$
  set $r_k \leftarrow \beta r_{k-1} + u_k$
  set $d_k \leftarrow v_k + P_k r_k$
  set $x_{k+1} \leftarrow x_k + \alpha d_k$
end for

For our analysis of Algorithm 2, we make the following assumption.

Assumption 3.1. *There exists an open convex set $\mathcal{X} \subseteq \mathbb{R}^n$ containing the iterates of any run of Algorithm 2 over which the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below by $f_{\inf} \in \mathbb{R}$, and over which the constraint function $c : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable and bounded in ℓ_2-norm in*

the sense that there exists $\kappa_c \in \mathbb{R}_{>0}$ *such that* $\|c(x)\|_2 \leq \kappa_c$ *for all* $x \in \mathcal{X}$*. In addition, there exist constants* $\kappa_{\nabla f} \in \mathbb{R}_{>0}$*,* $\kappa_J \in \mathbb{R}_{>0}$*,* $L_{\nabla f} \in \mathbb{R}_{>0}$*,* $\hat{L}_J \in \mathbb{R}_{>0}$*, and* $\sigma_{\min} \in \mathbb{R}_{>0}$ *such that one has that*

$$\begin{aligned}
\|\nabla f(x)\|_2 &\leq \kappa_{\nabla f} && \text{for all } \ x \in \mathcal{X}, \\
\|J(x)\|_2 &\leq \kappa_J && \text{for all } \ x \in \mathcal{X}, \\
\|\nabla f(x) - \nabla f(\bar{x})\|_2 &\leq L_{\nabla f}\|x - \bar{x}\|_2 && \text{for all } \ (x, \bar{x}) \in \mathcal{X} \times \mathcal{X}, \\
\|J(x) - J(\bar{x})\|_2 &\leq \hat{L}_J\|x - \bar{x}\|_2 && \text{for all } \ (x, \bar{x}) \in \mathcal{X} \times \mathcal{X}, \\
\text{and } \ \sigma_1(J(x)) &\geq \sigma_{\min} && \text{for all } \ x \in \mathcal{X},
\end{aligned}$$

where $\sigma_1(\cdot)$ *yields the smallest singular value of its matrix argument. Furthermore,*

$$\mathbb{E}[P_k g_k | \mathcal{F}_k] = P_k \nabla f(x_k) \ \ \text{for all } \ k \in \mathbb{N},$$

there exists a constant $M \in \mathbb{R}$ *such that*

$$\mathbb{E}[\|P_k(g_k - \nabla f(x_k))\|_2^2 | \mathcal{F}_k] \leq M^2 \ \ \text{for all } \ k \in \mathbb{N},$$

and there exist pairs of constants $(\rho_{\min}, \rho_{\max}) \in (0, 1] \times (0, 1]$ *and* $(h_{\min}, h_{\max}) \in \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ *such that* $\rho_{\min} \leq \rho_k \leq \rho_{\max}$ *and* $h_{\min} \leq h_k \leq h_{\max}$ *for all* $k \in \mathbb{N}$*.*

It is well known (e.g., see [3]) that, under Assumption 3.1, one has

$$f(\bar{x}) - f(x) \leq \nabla f(x)^T(\bar{x} - x) + \tfrac{1}{2}L_{\nabla f}\|\bar{x} - x\|_2^2 \ \ \text{for all } \ (x, \bar{x}) \in \mathcal{X} \times \mathcal{X}, \tag{9}$$

and, using norm inequalities, there exists $L_J \in \mathbb{R}_{>0}$ such that

$$\|c(\bar{x})\|_1 \leq \|c(x) + J(x)(\bar{x} - x)\|_1 + \tfrac{1}{2}L_J\|\bar{x} - x\|_2^2 \ \ \text{for all } \ (x, \bar{x}) \in \mathcal{X} \times \mathcal{X}. \tag{10}$$

Next, let us show that an important Lipschitz continuity property holds.

Lemma 3.1. *There exists* $L_{P\nabla f} \in \mathbb{R}_{>0}$ *such that for all* $(x, \bar{x}) \in \mathcal{X} \times \mathcal{X}$ *one has*

$$\|P(x)\nabla f(x) - P(\bar{x})\nabla f(\bar{x})\|_2 \leq L_{P\nabla f}\|x - \bar{x}\|_2.$$

Proof. Proof. Consider arbitrary $x \in \mathcal{X}$. Under Assumption 3.1, the pseudoinverse of $J(x)$ is a right inverse and $J(x)^\dagger := J(x)^T(J(x)J(x)^T)^{-1}$, so $P(x) = I - J(x)^\dagger J(x)$. Also, $P(x)$ is symmetric, and it is well known that $\|J(x)^\dagger\|_2 = \|J(x)^T(J(x)J(x)^T)^{-1}\|_2 \leq \frac{1}{\sigma_{\min}}$ for any $x \in \mathcal{X}$. Thus, for any $(x, \bar{x}) \in \mathcal{X} \times \mathcal{X}$, one has

$$\begin{aligned}
\|P(x) - P(\bar{x})\|_2 &= \|P(x)(I - P(\bar{x})) - (I - P(x))P(\bar{x})\|_2 \\
&= \|P(x)^T(I - P(\bar{x}))^T - (I - P(x))P(\bar{x})\|_2 \\
&= \|P(x)^T J(\bar{x})^T(J(\bar{x})^\dagger)^T - J(x)^\dagger J(x)P(\bar{x})\|_2 \\
&= \|P(x)^T(J(\bar{x}) - J(x))^T(J(\bar{x})^\dagger)^T - J(x)^\dagger(J(x) - J(\bar{x}))P(\bar{x})\|_2 \\
&\leq \|P(x)\|_2\|J(\bar{x}) - J(x)\|_2\|J(\bar{x})^\dagger\|_2 + \|J(x)^\dagger\|_2\|J(x) - J(\bar{x})\|_2\|P(\bar{x})\|_2 \\
&\leq (\|J(\bar{x})^\dagger\|_2 + \|J(x)^\dagger\|_2)\|J(x) - J(\bar{x})\|_2 \\
&\leq \frac{2\hat{L}_J}{\sigma_{\min}}\|x - \bar{x}\|_2.
\end{aligned}$$

Therefore, for any $(x, \bar{x}) \in \mathcal{X} \times \mathcal{X}$, one has

$$\begin{aligned}
\|P(x)\nabla f(x) - P(\bar{x})\nabla f(\bar{x})\|_2 &= \|P(x)\nabla f(x) - P(x)\nabla f(\bar{x}) + P(x)\nabla f(\bar{x}) - P(\bar{x})\nabla f(\bar{x})\|_2 \\
&\leq \|P(x)\|_2\|\nabla f(x) - \nabla f(\bar{x})\|_2 + \|P(x) - P(\bar{x})\|_2\|\nabla f(\bar{x})\|_2 \\
&\leq \left(L_{\nabla f} + \frac{2\hat{L}_J\kappa_{\nabla f}}{\sigma_{\min}}\right)\|x - \bar{x}\|_2 =: L_{P\nabla f}\|x - \bar{x}\|_2,
\end{aligned}$$

which completes the proof. □ □

Next we state a couple of bounds pertaining to finite series.

Lemma 3.2. *Given $\beta \in (0,1)$ and $K \in \mathbb{N}$, one has that*

$$\sum_{k=0}^{K-1} \beta^k = \frac{1-\beta^K}{1-\beta} \leq \frac{1}{1-\beta}, \quad \sum_{k=0}^{K-1} \beta^k k \leq \frac{\beta}{(1-\beta)^2}, \quad \text{and} \quad \sum_{k=0}^{K-1} \beta^k k^2 \leq \frac{\beta(1+\beta)}{(1-\beta)^3}.$$

Proof. Proof. Each bound is straightforward to verify; e.g., for the second, see [10, Lemma B.2]. □

We also state the following lemma, which follows easily from our prior observations under Assumption 3.1. Here and throughout the remainder of our analyses, we define $\phi : \mathbb{R}^n \times \mathbb{R}_{>0} \to \mathbb{R}$ by $\phi(x,\tau) = \tau f(x) + \|c(x)\|_1$.

Lemma 3.3. *For all $k \in \mathbb{N}$, it follows that*

$$\|v_k\|_2 = \|\rho_k J_k^T (J_k J_k^T)^{-1} c_k\|_2 \leq \rho_k \kappa_c \sigma_{\min}^{-1}, \tag{11}$$

$$\mathbb{E}[\|u_k\|_2^2 | \mathcal{F}_k] = h_k^{-2} \mathbb{E}[\|P_k g_k\|_2^2 | \mathcal{F}_k] \leq h_k^{-2}(\kappa_{\nabla f}^2 + M^2). \tag{12}$$

In addition, for any $\tau \in \mathbb{R}_{>0}$, it follows for all $k \in \mathbb{N}$ that

$$\phi(x_k + \alpha d_k, \tau) - \phi(x_k, \tau) \leq \tau\alpha \nabla f(x_k)^T d_k + \|c_k + \alpha J_k d_k\|_1 - \|c_k\|_1 + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\|d_k\|_2^2. \tag{13}$$

Proof. Proof. Consider arbitrary $k \in \mathbb{N}$. The bound (11) (respectively, (12)) follows from the definition of v_k (respectively, u_k) in the algorithm and Assumption 3.1. In addition, under Assumption 3.1, one has that

$$\begin{aligned}
&\phi(x_k + \alpha d_k, \tau) - \phi(x_k, \tau) \\
&= \tau(f(x_k + \alpha d_k) - f(x_k)) + \|c(x_k + \alpha d_k)\|_1 - \|c_k\|_1 \\
&\leq \tau(\alpha \nabla f(x_k)^T d_k + \tfrac{1}{2}\alpha^2 L_{\nabla f}\|d_k\|_2^2) + \|c_k + \alpha J_k d_k\|_1 + \tfrac{1}{2}\alpha^2 L_J \|d_k\|_2^2 - \|c_k\|_1 \\
&= \tau\alpha \nabla f(x_k)^T d_k + \|c_k + \alpha J_k d_k\|_1 - \|c_k\|_1 + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\|d_k\|_2^2,
\end{aligned}$$

which completes the proof. □ □

For our next lemmas, first observe that for all $k \in \mathbb{N}$ one has that

$$r_k = u_k + \beta r_{k-1} = u_k + \beta(u_{k-1} + \beta r_{k-2}) = \sum_{i=0}^{k-1} \beta^i u_{k-i} = \sum_{i=1}^{k} \beta^{k-i} u_i. \tag{14}$$

Lemma 3.4. *For all $k \in \mathbb{N}$, it follows that*

$$\mathbb{E}[\|r_k\|_2^2] \leq \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)^2}.$$

Proof. Proof. Consider arbitrary $k \in \mathbb{N}$. By (14), Jensen's inequality, and Lemmas 3.2 and 3.3 one finds that

$$\begin{aligned}
\mathbb{E}[\|r_k\|_2^2] = \mathbb{E}\left[\left\|\sum_{i=0}^{k-1} \beta^i u_{k-i}\right\|_2^2\right] &= \mathbb{E}\left[\left(\sum_{i=0}^{k-1} \beta^i\right)^2 \left\|\frac{\sum_{i=0}^{k-1} \beta^i u_{k-i}}{\sum_{i=0}^{k-1} \beta^i}\right\|_2^2\right] \\
&\leq \left(\sum_{i=0}^{k-1} \beta^i\right)^2 \frac{\sum_{i=0}^{k-1} \beta^i \mathbb{E}[\|u_{k-i}\|_2^2]}{\sum_{i=0}^{k-1} \beta^i} = \left(\sum_{i=0}^{k-1} \beta^i\right) \sum_{i=0}^{k-1} \beta^i \mathbb{E}[\|u_{k-i}\|_2^2] \\
&\leq \frac{1}{1-\beta} \sum_{i=0}^{k-1} \beta^i h_k^{-2}(\kappa_{\nabla f}^2 + M^2) \leq \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)^2},
\end{aligned}$$

as desired. □ □

Next, we bound the expected value of the inner product between the true gradient of the objective function and the search direction in each iteration.

Lemma 3.5. *For all $k \in \mathbb{N}$, it follows that*

$$\begin{aligned}\mathbb{E}[\nabla f(x_k)^T d_k] \leq & -\sum_{i=0}^{k-1} \beta^i h_{\max}^{-1} \mathbb{E}[\|P_{k-i}\nabla f(x_{k-i})\|_2^2] \\ & + \frac{\rho_{\max}\kappa_{\nabla f}\mathbb{E}[\|c_k\|_2]}{\sigma_{\min}} + \frac{\beta L_{P\nabla f}\alpha}{(1-\beta)^2}\left(\frac{\rho_{\max}^2\kappa_c^2(1-\beta)}{2\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2(1-\beta)}\right).\end{aligned}$$

Proof. Proof. Consider arbitrary $k \in \mathbb{N}$. By the definition of d_k, $P_k = P_k^T$, and (14), one finds that

$$\begin{aligned}\nabla f(x_k)^T d_k &= \nabla f(x_k)^T (v_k + P_k r_k) \\ &= \nabla f(x_k)^T \left(v_k + P_k \sum_{i=0}^{k-1} \beta^i u_{k-i}\right) \\ &= -\rho_k \nabla f(x_k)^T J_k^T (J_k J_k^T)^{-1} c_k + \sum_{i=0}^{k-1} \beta^i \nabla f(x_k)^T P_k u_{k-i} \\ &= -\rho_k \nabla f(x_k)^T J_k^T (J_k J_k^T)^{-1} c_k \\ &\quad + \sum_{i=0}^{k-1} \beta^i \nabla f(x_{k-i})^T P_{k-i}^T u_{k-i} + \sum_{i=1}^{k-1} \beta^i (P_k \nabla f(x_k) - P_{k-i}\nabla f(x_{k-i}))^T u_{k-i}. \end{aligned} \tag{15}$$

With respect to the first term in (15), it follows by Assumption 3.1 and Lemma 3.3 (specifically, (11)) that

$$-\rho_k \nabla f(x_k)^T J_k^T (J_k J_k^T)^{-1} c_k \leq \frac{\rho_{\max}\kappa_{\nabla f}\|c_k\|_2}{\sigma_{\min}}. \tag{16}$$

With respect to the second term in (15), it follows with Assumption 3.1 that for all $i \in \{0, \ldots, k-1\}$ one has

$$\begin{aligned}\mathbb{E}[\nabla f(x_{k-i})^T P_{k-i}^T u_{k-i} | \mathcal{F}_{k-i}] &= \mathbb{E}[-h_{k-i}^{-1} \nabla f(x_{k-i})^T P_{k-i}^T P_{k-i} g_{k-i} | \mathcal{F}_{k-i}] \\ &= -h_{k-i}^{-1} \nabla f(x_{k-i})^T P_{k-i}^T P_{k-i} \nabla f(x_{k-i}) \\ &= -h_{k-i}^{-1} \|P_{k-i}\nabla f(x_{k-i})\|_2^2 \leq -h_{\max}^{-1} \|P_{k-i}\nabla f(x_{k-i})\|_2^2. \end{aligned} \tag{17}$$

With respect to the third term in (15), it follows by Lemma 3.1, Jensen's inequality, Lemma 3.3 (specifically, inequality (11)), and v_j and $P_j r_j$ being orthogonal for any $j \in \mathbb{N}$ that for all $i \in \{1, \ldots, k-1\}$ one has

$$\begin{aligned}\|P_k \nabla f(x_k) - P_{k-i}\nabla f(x_{k-i})\|_2^2 &\leq L_{P\nabla f}^2 \|x_k - x_{k-i}\|_2^2 \\ &= L_{P\nabla f}^2 \left\| \alpha \sum_{j=k-i}^{k-1} (v_j + P_j r_j) \right\|_2^2 \\ &\leq L_{P\nabla f}^2 i\alpha^2 \left(\sum_{j=k-i}^{k-1} \|v_j\|_2^2 + \sum_{j=k-i}^{k-1} \|P_j r_j\|_2^2 \right) \\ &\leq L_{P\nabla f}^2 i\alpha^2 \left(i\rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} + \sum_{j=k-i}^{k-1} \|P_j r_j\|_2^2 \right). \end{aligned} \tag{18}$$

Consequently, along with the Cauchy-Schwarz inequality and since Young's inequality implies that $|ab| \leq \frac{\lambda}{2}|a|^2 + \frac{1}{2\lambda}|b|^2$ for any $(a,b) \in \mathbb{R} \times \mathbb{R}$ and $\lambda \in \mathbb{R}_{>0}$, one finds with $\lambda = \frac{1-\beta}{L_{P\nabla f} i\alpha}$ that

$$\begin{aligned}
&\sum_{i=1}^{k-1} \beta^i (P_k \nabla f(x_k) - P_{k-i} \nabla f(x_{k-i}))^T u_{k-i} \\
&\leq \sum_{i=1}^{k-1} \beta^i \|P_k \nabla f(x_k) - P_{k-i} \nabla f(x_{k-i})\|_2 \|u_{k-i}\|_2 \\
&\leq \sum_{i=1}^{k-1} \beta^i \left(\frac{1-\beta}{2L_{P\nabla f} i\alpha} \|P_k \nabla f(x_k) - P_{k-i} \nabla f(x_{k-i})\|_2^2 + \frac{L_{P\nabla f} i\alpha}{2(1-\beta)} \|u_{k-i}\|_2^2 \right) \\
&\leq \sum_{i=1}^{k-1} \beta^i L_{P\nabla f} \alpha \left(\frac{1-\beta}{2} \left(i\rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} + \sum_{j=k-i}^{k-1} \|P_j r_j\|_2^2 \right) + \frac{i}{2(1-\beta)} \|u_{k-i}\|_2^2 \right).
\end{aligned}$$

Taking total expectation and using Lemma 3.2, Lemma 3.3 (specifically, (12)), and Lemma 3.4, one finds that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{k-1} \beta^i (P_k \nabla f(x_k) - P_{k-i} \nabla f(x_{k-i}))^T u_{k-i} \right] \\
&\leq \sum_{i=1}^{k-1} \beta^i L_{P\nabla f} \alpha \left(\frac{1-\beta}{2} \left(i\rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} + \sum_{j=k-i}^{k-1} \mathbb{E}\left[\|P_j r_j\|_2^2 \right] \right) + \frac{i}{2(1-\beta)} \mathbb{E}\left[\|u_{k-i}\|_2^2 \right] \right) \\
&\leq \sum_{i=1}^{k-1} \beta^i L_{P\nabla f} \alpha i \left(\frac{1-\beta}{2} \left(\rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)^2} \right) + \frac{\kappa_{\nabla f}^2 + M^2}{2h_{\min}^2 (1-\beta)} \right) \\
&= L_{P\nabla f} \alpha \left(\frac{\rho_{\max}^2 \kappa_c^2 (1-\beta)}{2\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)} \right) \sum_{i=1}^{k-1} \beta^i i \leq \frac{\beta L_{P\nabla f} \alpha}{(1-\beta)^2} \left(\frac{\rho_{\max}^2 \kappa_c^2 (1-\beta)}{2\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)} \right). \qquad (19)
\end{aligned}$$

Taking expectation on both sides of (15) and using (16), (17), and (19), the proof is complete. □ □

Now we bound the expected reduction in the merit function between two consecutive iterations.

Lemma 3.6. *For any $\tau \in \mathbb{R}_{>0}$, it follows for all $k \in \mathbb{N}$ that*

$$\begin{aligned}
&\mathbb{E}[\phi(x_{k+1}) - \phi(x_k)] \\
&\quad \leq -\alpha \left(\tau \sum_{i=0}^{k-1} \beta^i h_{\max}^{-1} \mathbb{E}[\|P_{k-i} \nabla f(x_{k-i})\|_2^2] + \left(1 - \frac{\tau \kappa_{\nabla f} \rho_{\max}}{\sigma_{\min} \rho_{\min}} \right) \rho_{\min} \mathbb{E}[\|c_k\|_1] \right) + \alpha^2 C, \\
&\quad \text{where } \; C := \frac{\beta L_{P\nabla f} \tau}{1-\beta} \left(\frac{\rho_{\max}^2 \kappa_c^2}{2\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)^2} \right) + \tfrac{1}{2}(\tau L_{\nabla f} + L_J) \left(\frac{\rho_{\max}^2 \kappa_c^2}{\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2 (1-\beta)^2} \right).
\end{aligned}$$

Proof. Proof. Consider arbitrary $\tau \in \mathbb{R}_{>0}$ and $k \in \mathbb{N}$. By Lemma 3.3, $\alpha \in (0,1]$, $\rho_k \in (0,1]$, and

$$J_k d_k = J_k v_k + J_k P_k r_k = J_k v_k = -\rho_k J_k J_k^T (J_k J_k^T)^{-1} c_k = -\rho_k c_k,$$

one has that

$$\begin{aligned}
\phi(x_{k+1}) - \phi(x_k) &\leq \tau \alpha \nabla f(x_k)^T d_k + \|c_k + \alpha J_k d_k\|_1 - \|c_k\|_1 + \tfrac{1}{2}\alpha^2 (\tau L_{\nabla f} + L_J) \|d_k\|_2^2 \\
&= \tau \alpha \nabla f(x_k)^T d_k - \alpha \rho_k \|c_k\|_1 + \tfrac{1}{2}\alpha^2 (\tau L_{\nabla f} + L_J) \|d_k\|_2^2. \qquad (20)
\end{aligned}$$

With respect to the last term, it follows from Lemma 3.3, Lemma 3.4, and $v_k^T P_k r_k = 0$ that

$$\begin{aligned}\tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\mathbb{E}[\|d_k\|_2^2] =& \ \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\mathbb{E}[\|v_k + P_k r_k\|_2^2] \\ \leq& \ \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\left(\frac{\rho_{\max}^2\kappa_c^2}{\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2(1-\beta)^2}\right).\end{aligned}$$

Now taking expectation on both sides of (20), and using Lemma 3.5 and basic norm inequalities, one finds

$$\begin{aligned}&\mathbb{E}[\phi(x_{k+1}) - \phi(x_k)] \\ \leq& \ -\tau\alpha\sum_{i=0}^{k-1}\beta^i h_{\max}^{-1}\mathbb{E}[\|P_{k-i}\nabla f(x_{k-i})\|_2^2] - \alpha\rho_{\min}\mathbb{E}[\|c_k\|_1] + \frac{\tau\alpha\rho_{\max}\kappa_{\nabla f}\mathbb{E}[\|c_k\|_1]}{\sigma_{\min}} \\ &+ \frac{\beta L_{P\nabla f}\alpha^2\tau}{(1-\beta)^2}\left(\frac{\rho_{\max}^2\kappa_c^2(1-\beta)}{2\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2(1-\beta)}\right) + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\left(\frac{\rho_{\max}^2\kappa_c^2}{\sigma_{\min}^2} + \frac{\kappa_{\nabla f}^2 + M^2}{h_{\min}^2(1-\beta)^2}\right),\end{aligned}$$

which completes the proof. □ □

Theorem 3.1. *Suppose that Assumption 3.1 holds and define*

$$\tau := \frac{\sigma_{\min}\rho_{\min}}{\sigma_{\min}\rho_{\min} + \kappa_{\nabla f}\rho_{\max}} \implies 1 - \frac{\tau\kappa_{\nabla f}\rho_{\max}}{\sigma_{\min}\rho_{\min}} = \tau. \tag{21}$$

Then, for any $K \in \mathbb{N}$ and with $C \in \mathbb{R}_{>0}$ defined in Lemma 3.6, it follows that

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[h_{\max}^{-1}\|P_k\nabla f(x_k)\|_2^2 + \rho_{\min}\|c_k\|_1] \leq \frac{\phi(x_1) - \tau f_{\inf}}{\alpha\tau K} + \frac{\alpha C}{\tau}. \tag{22}$$

Proof. Proof. Consider arbitrary $k \in \mathbb{N}$. Given τ defined by (21), it follows from Lemma 3.6 that

$$\mathbb{E}[\phi(x_{k+1}) - \phi(x_k)] \leq -\alpha\tau\left(\sum_{i=0}^{k-1}\beta^i h_{\max}^{-1}\mathbb{E}[\|P_{k-i}\nabla f(x_{k-i})\|_2^2] + \rho_{\min}\mathbb{E}[\|c_k\|_1]\right) + \alpha^2 C.$$

Summing this inequality over $k \in \{1, \ldots, K\}$ and using $\phi(x) \geq \tau f_{\inf}$ for all $x \in \mathcal{X}$ yields

$$\frac{1}{K}\left(\sum_{k=1}^{K}\sum_{i=0}^{k-1}\beta^i h_{\max}^{-1}\mathbb{E}[\|P_{k-i}\nabla f(x_{k-i})\|_2^2] + \sum_{k=1}^{K}\rho_{\min}\mathbb{E}[\|c_k\|_1]\right) \leq \frac{\phi(x_1) - \tau f_{\inf}}{\alpha\tau K} + \frac{\alpha C}{\tau}. \tag{23}$$

With respect to the first term in the parentheses on the left-hand side, one finds that

$$\begin{aligned}\sum_{k=1}^{K}\sum_{i=0}^{k-1}\beta^i\mathbb{E}[\|P_{k-i}\nabla f(x_{k-i})\|_2^2] &= \sum_{k=1}^{K}\sum_{i=1}^{k}\beta^{k-i}\mathbb{E}[\|P_i\nabla f(x_i)\|_2^2] \\ &= \sum_{k=1}^{K}\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2]\sum_{i=0}^{K-k}\beta^i \\ &= \sum_{k=1}^{K}\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2]\frac{1-\beta^{K-k+1}}{1-\beta} \\ &\geq \sum_{k=1}^{K}\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2].\end{aligned}$$

This inequality, along with (23), completes the proof. □ □

Theorem 3.1 shows that the average expected combination of the squared norm of the projected gradient and ℓ_1-norm constraint violation over K iterations decreases to $\alpha C/\tau$ at a rate of $\mathcal{O}(1/K)$. Indeed, the upper bound on the average expected combination can be made as small as desired: with $\{\rho_k\}$, $\{h_k\}$, and β fixed, one can choose α small enough such that $\alpha C/\tau$ is as small as desired, then choose K sufficiently large such that the first term on the right-hand side of (22) is as small as desired. In addition, $\rho_{\min}$ and $h_{\max}$ can be chosen to achieve any desired balance between $\mathbb{E}[\|P_k \nabla f(x_k)\|_2^2]$ and $\mathbb{E}[\|c_k\|_1]$ in the left-hand side of (22).

3.2 Projected Stochastic Adam SQP

Next, we consider an extension of the Adam method to the setting of equality-constrained optimization. Specifically, we propose Algorithm 3, where we note that for any equal-length real vectors a and b we use $a \circ b$ to denote their component-wise product, we use e to denote a vector of all ones whose length is determined by the context in which it appears, and for any real vector v we use $\operatorname{diag}(v)$ to denote a diagonal matrix whose diagonal components are those of v (in order).

We emphasize that, like the variant of Adam that is analyzed in [10], Algorithm 3 involves a modified bias correction term in order to guarantee that a certain step size sequence (see the sequences $\{\eta_k\}$ and $\{\alpha_k\}$ below) is monotonically nondecreasing as $k \to \infty$. As discussed in [10], this variant for the unconstrained setting regularly yields comparable performance with the original Adam method. We borrow this modified bias correction idea in order to model our analysis on that in [10].

Algorithm 3 Projected Stochastic Adam SQP

Require: $x_1 \in \mathbb{R}^n$, $\{\rho_k\}$ and $\{h_k\}$ with $\rho_k \in (0,1]$ and $h_k \in \mathbb{R}_{>0}$ for all $k \in \mathbb{N}$, $\beta_1 \in [0,1)$, $\beta_2 \in (\beta_1, 1)$, $\alpha \in (0,1]$, and $\epsilon \in \mathbb{R}_{>0}$

set $r_0 \leftarrow 0 \in \mathbb{R}^n$
set $s_0 \leftarrow 0 \in \mathbb{R}^n$
for all $k \in \mathbb{N}$ **do**
  compute a stochastic gradient estimate $g_k \approx \nabla f(x_k)$
  compute $v_k \leftarrow -\rho_k J_k^T \left(J_k J_k^T\right)^{-1} c_k$
  compute $u_k \leftarrow -h_k^{-1} P_k g_k$, where $P_k := I - J_k^T \left(J_k J_k^T\right)^{-1} J_k$
  set $r_k \leftarrow \beta_1 r_{k-1} + u_k$ (first momentum)
  set $s_k \leftarrow \beta_2 s_{k-1} + u_k \circ u_k$ (second momentum)
  set $\eta_k \leftarrow \frac{(1-\beta_1)\sqrt{1-\beta_2^k}}{\sqrt{1-\beta_2}}$ (bias correction)
  set $d_k \leftarrow v_k + \eta_k P_k \operatorname{diag}(s_k + \epsilon e)^{-1/2} r_k$
  Set $x_{k+1} \leftarrow x_k + \alpha d_k$
end for

For our analysis of Algorithm 3, we make the following assumption.

Assumption 3.2. *With respect to Algorithm 3, the conditions of Assumption 3.1 hold and, in addition,*

$$\|u_k\|_\infty \equiv \|h_k^{-1} P_k g_k\|_\infty \leq \sqrt{\frac{M^2 + \kappa_{\nabla f}^2}{h_k^2} - \epsilon} \quad \textit{for all} \quad k \in \mathbb{N}.$$

Before commencing our analysis, let us define a few quantities to simplify our expressions. Let us also note that for any vector defined by the algorithm, we use a second subscript to denote a component index; e.g., for any (k,i) we use $s_{k,i}$ to denote the ith component of the vector s_k. Similarly, for the product $P_k g_k$, we use $[P_k g_k]_i$ to denote its ith component. For the sake of simplicity, let us define the step size for u_k as

$$\alpha_k := \alpha \eta_k = \frac{\alpha(1-\beta_1)\sqrt{1-\beta_2^k}}{\sqrt{1-\beta_2}} \quad \text{for all} \quad k \in \mathbb{N},$$

and observe that it is nondecreasing as $k \to \infty$. In addition, let us define for all $k \in \mathbb{N}$ the vectors t_k and $\tilde{t}_k$, where for all $i \in \{1, \ldots, n\}$ the ith component of each vector is given by

$$t_{k,i} := \frac{r_{k,i}}{\sqrt{s_{k,i} + \epsilon}}, \quad \text{and} \quad \tilde{t}_{k,i} := \frac{u_{k,i}}{\sqrt{s_{k,i} + \epsilon}} = -\frac{[P_k g_k]_i}{h_k \sqrt{s_{k,i} + \epsilon}},$$

respectively. Let us also note that for all (k, i) and $j \in \{1, \ldots, k\}$ one has that

$$s_{k,i} = \sum_{l=1}^{k} \beta_2^{k-l} h_l^{-2} [P_l g_l]_i^2 = \beta_2^j s_{k-j,i} + \sum_{l=k-j+1}^{k} \beta_2^{k-l} h_l^{-2} [P_l g_l]_i^2,$$

and for all (k, i) and $j \in \{1, \ldots, k\}$ let us define the related quantity

$$\tilde{s}_{k,j,i} = \beta_2^j s_{k-j,i} + \mathbb{E}\left[\sum_{l=k-j+1}^{k} \beta_2^{k-l} h_l^{-2} [P_l g_l]_i^2 \middle| \mathcal{F}_{k-j+1} \right]. \tag{24}$$

Also, similarly as for (14) for the heavy-ball method, one finds here for all $k \in \mathbb{N}$ that

$$r_k = u_k + \beta_1 r_{k-1} = u_k + \beta_1 (u_{k-1} + \beta_1 r_{k-2}) = \sum_{i=0}^{k-1} \beta_1^i u_{k-i} = \sum_{i=1}^{k} \beta_1^{k-i} u_i. \tag{25}$$

We begin with two technical lemmas whose proofs can be found in [10].

Lemma 3.7. *Let $(\beta_1, \beta_2, \epsilon)$ be given by Algorithm 3 and let $\{a_k\}$ be any sequence of real numbers. For any $k \in \mathbb{N}$, with $b_k := \sum_{j=1}^{k} \beta_2^{k-j} a_j^2$ and $q_k := \sum_{j=1}^{k} \beta_1^{k-j} a_j$, one has that*

$$\sum_{j=1}^{k} \frac{q_j^2}{b_j + \epsilon} \leq \frac{1}{(1-\beta_1)(1-\frac{\beta_1}{\beta_2})} \left(\log\left(1 + \frac{b_k}{\epsilon}\right) - k \log(\beta_2) \right)$$

$$\text{and} \quad \sum_{j=1}^{k} \frac{a_j^2}{b_j + \epsilon} \leq \log\left(1 + \frac{b_k}{\epsilon}\right) - k \log(\beta_2).$$

Proof. Proof. See [10, Lemmas 5.2 and A.2]. □

Lemma 3.8. *For any $k \in \mathbb{N}$ and $\beta \in (0, 1)$, it follows that*

$$\sum_{j=0}^{k-1} \beta^j \sqrt{j+1} \leq \frac{2}{(1-\beta)^{3/2}} \quad \text{and} \quad \sum_{j=0}^{k-1} \beta^j \sqrt{j}(j+1) \leq \frac{4\beta}{(1-\beta)^{5/2}}.$$

Proof. Proof. See [10, Lemmas A.3 and A.4]. □

Now, similarly as in Lemma 3.5 for the heavy-ball method in the previous subsection, we bound the expected inner product between the true gradient of the objective function and the search direction in each iteration.

Lemma 3.9. *For all $k \in \mathbb{N}$, it follows that*

$$\begin{aligned} &\mathbb{E}[\nabla f(x_k)^T d_k] \\ &\leq -\frac{\eta_k}{2h_{\max}} \sum_{j=0}^{k-1} \beta_1^j \mathbb{E}[(P_{k-j} \nabla f(x_{k-j}))^T \operatorname{diag}(\tilde{s}_{k,j+1} + \epsilon e)^{-1/2} P_{k-j} \nabla f(x_{k-j})] \end{aligned}$$

$$+\frac{\rho_{\max}\kappa_{\nabla f}\mathbb{E}[\|c_k\|_2]}{\sigma_{\min}}+\frac{3\eta_k\sqrt{M^2+\kappa_{\nabla f}^2}h_{\max}}{h_{\min}\sqrt{1-\beta_1}}\sum_{j=0}^{k-1}\left(\frac{\beta_1}{\beta_2}\right)^j\sqrt{j+1}\mathbb{E}[\|\tilde{t}_{k-j}\|_2^2]$$
$$+\eta_k L_{P\nabla f}^2\alpha^2\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2}\frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\frac{\beta_1(1+\beta_1)}{(1-\beta_1)^3}$$
$$+\eta_k L_{P\nabla f}^2\alpha_k^2\frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{j=1}^{k-1}\mathbb{E}[\|t_{k-j}\|_2^2]\sum_{l=j}^{k-1}\beta_1^l\sqrt{l}.$$

Proof. Proof. Consider arbitrary $k\in\mathbb{N}$. By the definition of d_k, $P_k=P_k^T$, and (25), one finds that

$$\begin{aligned}\nabla f(x_k)^T d_k &= \nabla f(x_k)^T(v_k+\eta_k P_k\operatorname{diag}(s_k+\epsilon e)^{-1/2}r_k)\\ &= \nabla f(x_k)^T v_k-\eta_k\sum_{j=0}^{k-1}\beta_1^j h_{k-j}^{-1}\nabla f(x_k)^T P_k\operatorname{diag}(s_k+\epsilon e)^{-1/2}P_{k-j}g_{k-j}\\ &= \underbrace{\nabla f(x_k)^T v_k}_{A}\underbrace{-\eta_k\sum_{j=0}^{k-1}\beta_1^j h_{k-j}^{-1}(P_{k-j}\nabla f(x_{k-j}))^T\operatorname{diag}(s_k+\epsilon e)^{-1/2}P_{k-j}g_{k-j}}_{B}\\ &\quad\underbrace{-\eta_k\sum_{j=0}^{k-1}\beta_1^j h_{k-j}^{-1}\left(P_k\nabla f(x_k)-P_{k-j}\nabla f(x_{k-j})\right)^T\operatorname{diag}(s_k+\epsilon e)^{-1/2}P_{k-j}g_{k-j}}_{C}.\end{aligned}\tag{26}$$

Term A satisfies (16). With respect to term B, one finds for any $j\in\{1,\dots,k-1\}$ that

$$\begin{aligned}(P_{k-j}\nabla f(x_{k-j}))^T\operatorname{diag}(s_k+\epsilon e)^{-1/2}P_{k-j}g_{k-j} &= \underbrace{(P_{k-j}\nabla f(x_{k-j}))^T\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2}P_{k-j}g_{k-j}}_{B_1}\\ &\quad+\underbrace{(P_{k-j}\nabla f(x_{k-j}))^T\left(\operatorname{diag}(s_k+\epsilon e)^{-1/2}-\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2}\right)P_{k-j}g_{k-j}}_{B_2}.\end{aligned}$$

Thus, by the definition (24), one finds for B_1 that

$$\mathbb{E}[B_1]=\mathbb{E}[\mathbb{E}[B_1|\mathcal{F}_{k-j}]]=\mathbb{E}[(P_{k-j}\nabla f(x_{k-j}))^T\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2}P_{k-j}\nabla f(x_{k-j})],$$

and at the same time one finds for B_2 that, from [10, Pages 19–20, Eq(A.27)] and Assumption 3.2, one has

$$\begin{aligned}\mathbb{E}[|B_2|] &\le \frac{1}{2}\mathbb{E}[(P_{k-j}\nabla f(x_{k-j}))^T\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2}P_{k-j}\nabla f(x_{k-j})]\\ &\quad+\frac{2\sqrt{M^2+\kappa_{\nabla f}^2}}{h_{\min}\sqrt{1-\beta_1}\beta_2^j}\sqrt{j+1}\mathbb{E}[(P_{k-j}g_{k-j})^T\operatorname{diag}(s_{k-j}+\epsilon e)^{-1}P_{k-j}g_{k-j}]\\ &= \frac{1}{2}\mathbb{E}[(P_{k-j}\nabla f(x_{k-j}))^T\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2}P_{k-j}\nabla f(x_{k-j})]\\ &\quad+\frac{2\sqrt{M^2+\kappa_{\nabla f}^2}}{h_{\min}\sqrt{1-\beta_1}\beta_2^j}\sqrt{j+1}h_{k-j}^2\mathbb{E}[\|\tilde{t}_{k-j}\|_2^2].\end{aligned}$$

From the definitions of B_1 and B_2, one finds that

$$B=\eta_k\sum_{j=0}^{k-1}\beta_1^j h_{k-j}^{-1}(B_1+B_2)\ge\eta_k\sum_{j=0}^{k-1}\beta_1^j h_{k-j}^{-1}(B_1-|B_2|),$$

so taking expectation and employing the above equation for $\mathbb{E}[B_1]$ and bound for $\mathbb{E}[|B_2|]$ yields

$$\begin{aligned}\mathbb{E}[B] &\geq \eta_k \sum_{j=0}^{k-1} \beta_1^j h_{k-j}^{-1}(\mathbb{E}[B_1] - \mathbb{E}[|B_2|]) \\ &\geq \frac{\eta_k}{2h_{\max}} \sum_{j=0}^{k-1} \beta_1^j \mathbb{E}[(P_{k-j}\nabla f(x_{k-j}))^T \operatorname{diag}(\tilde{s}_{k,j+1} + \epsilon e)^{-1/2} P_{k-j}\nabla f(x_{k-j})] \\ &\quad - \frac{2\eta_k\sqrt{M^2+\kappa_{\nabla f}^2}h_{\max}}{h_{\min}\sqrt{1-\beta_1}} \sum_{j=0}^{k-1} \left(\frac{\beta_1}{\beta_2}\right)^j \sqrt{j+1}\mathbb{E}[\|\tilde{t}_{k-j}\|_2^2]. \end{aligned} \tag{27}$$

Now with respect to term C in (26), with Lemma 3.1, Jensen's inequality, Lemma 3.3 (specifically, inequality (11)), v_{k-j} and $P_{k-j}t_{k-j}$ being orthogonal for any $k-l \in \mathbb{N}$, and the fact that $\{\alpha_k\}$ is nondecreasing, one finds (similarly as in (18)) that for any $j \in \{1, \ldots, k-1\}$ one has

$$\begin{aligned}\|P_k\nabla f(x_k) - P_{k-j}\nabla f(x_{k-j})\|_2^2 &\leq L_{P\nabla f}^2 \|x_k - x_{k-j}\|_2^2 \\ &= L_{P\nabla f}^2 \left\|\sum_{l=1}^{j} (\alpha v_{k-l} + \alpha_{k-l}P_{k-l}t_{k-l})\right\|_2^2 \\ &\leq L_{P\nabla f}^2 j \left(\sum_{l=1}^{j} \|\alpha v_{k-l}\|_2^2 + \sum_{l=1}^{j} \|\alpha_{k-l}P_{k-l}t_{k-l}\|_2^2\right) \\ &\leq L_{P\nabla f}^2 j^2\alpha^2\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2} + L_{P\nabla f}^2 j\alpha_{k-1}^2 \sum_{l=1}^{j} \|t_{k-l}\|_2^2 \\ &\leq L_{P\nabla f}^2 j^2\alpha^2\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2} + L_{P\nabla f}^2 j\alpha_k^2 \sum_{l=1}^{j} \|t_{k-l}\|_2^2. \end{aligned}$$

At the same time, for the vector $\operatorname{diag}(s_k + \epsilon e)^{-1/2}P_{k-j}g_{k-j}$, one finds for all $i \in \{1, \ldots, n\}$ that

$$\begin{aligned}&\epsilon + s_{k,i} = \epsilon + \beta_2^j s_{k-j,i} + \sum_{l=k-j+1}^{k} h_l^{-2}\beta_2^{k-l}(P_l g_l)_i^2 \geq \epsilon + \beta_2^j s_{k-j,i} \geq \beta_2^j(\epsilon + s_{k-j,i}) \\ &\implies \frac{(P_{k-j}g_{k-j})_i^2}{\epsilon + s_{k,i}} \leq \frac{1}{\beta_2^j}\frac{(P_{k-j}g_{k-j})_i^2}{\epsilon + s_{k-j,i}} = \frac{h_{k-j}^2}{\beta_2^j}\frac{(P_{k-j}g_{k-j})_i^2}{h_{k-j}^2(\epsilon + s_{k-j,i})} = \frac{h_{k-j}^2}{\beta_2^j}\tilde{t}_{k-j,i}^2. \end{aligned}$$

Thus, along with the Cauchy-Schwarz inequality and since Young's inequality implies that $|ab| \leq \frac{\lambda}{2}|a|^2 + \frac{1}{2\lambda}|b|^2$ for any $(a,b) \in \mathbb{R}\times\mathbb{R}$ and $\lambda \in \mathbb{R}_{>0}$, one finds with $\lambda = \frac{h_{\min}\sqrt{1-\beta_1}}{2\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{j+1}}$ that

$$\begin{aligned}&(P_k\nabla f(x_k) - P_{k-j}\nabla f(x_{k-j}))^T \operatorname{diag}(s_k + \epsilon e)^{-1/2}P_{k-j}g_{k-j} \\ &\leq \|P_k\nabla f(x_k) - P_{k-j}\nabla f(x_{k-j})\|_2 \|\operatorname{diag}(s_k + \epsilon e)^{-1/2}P_{k-j}g_{k-j}\|_2 \\ &\leq \frac{h_{\min}\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{j+1}}\|P_k\nabla f(x_k) - P_{k-j}\nabla f(x_{k-j})\|_2^2 \\ &\quad + \frac{\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{j+1}}{h_{\min}\sqrt{1-\beta_1}}\|\operatorname{diag}(s_k + \epsilon e)^{-1/2}P_{k-j}g_{k-j}\|_2^2 \\ &\leq \frac{h_{\min}\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{j+1}}\left(L_{P\nabla f}^2 j^2\alpha^2\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2} + L_{P\nabla f}^2 j\alpha_k^2 \sum_{l=1}^{j} \|t_{k-l}\|_2^2\right) \end{aligned}$$

$$+ \frac{\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{j+1}}{h_{\min}\sqrt{1-\beta_1}} \frac{h_{k-j}^2}{\beta_2^j} \|\tilde{t}_{k-j}\|_2^2.$$

Thus, using Lemma 3.2, one finds that

$$\begin{aligned}
&\mathbb{E}[-C] \\
&\leq \mathbb{E}[|C|] = \eta_k \sum_{j=0}^{k-1} \beta_1^j h_{k-j}^{-1} \mathbb{E}\left[\left|(P_k \nabla f(x_k) - P_{k-j}\nabla f(x_{k-j}))^T \operatorname{diag}(s_k+\epsilon e)^{-1/2} P_{k-j} g_{k-j}\right|\right] \\
&\leq \eta_k L_{P\nabla f}^2 \alpha^2 \rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} \frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}} \sum_{j=0}^{k-1} \frac{\beta_1^j j^2}{\sqrt{j+1}} \\
&\quad + \eta_k L_{P\nabla f}^2 \alpha_k^2 \frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}} \sum_{j=0}^{k-1} \frac{\beta_1^j j}{\sqrt{j+1}} \sum_{l=1}^{j} \mathbb{E}[\|t_{k-l}\|_2^2] \\
&\quad + \frac{\eta_k \sqrt{M^2+\kappa_{\nabla f}^2} h_{\max}}{h_{\min}\sqrt{1-\beta_1}} \sum_{j=0}^{k-1} \left(\frac{\beta_1}{\beta_2}\right)^j \sqrt{j+1} \mathbb{E}[\|\tilde{t}_{k-j}\|_2^2] \\
&\leq \eta_k L_{P\nabla f}^2 \alpha^2 \rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} \frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}} \sum_{j=0}^{k-1} \beta_1^j j^2 + \eta_k L_{P\nabla f}^2 \alpha_k^2 \frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}} \sum_{j=0}^{k-1} \beta_1^j \sqrt{j} \sum_{l=1}^{j} \mathbb{E}[\|t_{k-l}\|_2^2] \\
&\quad + \frac{\eta_k \sqrt{M^2+\kappa_{\nabla f}^2} h_{\max}}{h_{\min}\sqrt{1-\beta_1}} \sum_{j=0}^{k-1} \left(\frac{\beta_1}{\beta_2}\right)^j \sqrt{j+1} \mathbb{E}[\|\tilde{t}_{k-j}\|_2^2] \\
&\leq \eta_k L_{P\nabla f}^2 \alpha^2 \rho_{\max}^2 \kappa_c^2 \sigma_{\min}^{-2} \frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}} \frac{\beta_1(1+\beta_1)}{(1-\beta_1)^3} \\
&\quad + \eta_k L_{P\nabla f}^2 \alpha_k^2 \frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}} \sum_{j=1}^{k-1} \mathbb{E}[\|t_{k-j}\|_2^2] \sum_{l=j}^{k-1} \beta_1^l \sqrt{l} \\
&\quad + \frac{\eta_k \sqrt{M^2+\kappa_{\nabla f}^2} h_{\max}}{h_{\min}\sqrt{1-\beta_1}} \sum_{j=0}^{k-1} \left(\frac{\beta_1}{\beta_2}\right)^j \sqrt{j+1} \mathbb{E}[\|\tilde{t}_{k-j}\|_2^2]. \qquad (28)
\end{aligned}$$

Combining (26) with the bounds (16), (27), and (28) completes the proof. □ □

Theorem 3.2. *Suppose that Assumptions 3.1 and 3.2 hold and define* $\tau \in \mathbb{R}_{>0}$ *by* (21) *along with*

$$G_1(\beta_1,\beta_2) := \frac{\tau\beta_1 L_{P\nabla f}^2}{(1-\beta_2)^{3/2}\sqrt{M^2+\kappa_{\nabla f}^2}}, \quad G_2(\beta_1,\beta_2) := \frac{(1-\beta_1)(\tau L_{\nabla f}+L_J)}{2(1-\beta_2)},$$
$$G_3(\beta_1,\beta_2) := \frac{\tau L_{P\nabla f}^2 \rho_{\max}^2 \kappa_c^2 \sqrt{1-\beta_1}\beta_1(1+\beta_1)}{4\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{1-\beta_2}(1-\beta_1)^2\sigma_{\min}^2}, \quad \textit{and} \quad G_4(\beta_1,\beta_2) := \frac{(\tau L_{\nabla f}+L_J)\rho_{\max}^2\kappa_c^2}{2\sigma_{\min}^2}.$$

Then, for any $K \in \mathbb{N}$*, it follows that*

$$\frac{1}{K}\sum_{k=1}^{K}\left(\frac{h_{\min}(1-\beta_1)}{2h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2] + \rho_{\min}\mathbb{E}[\|c_k\|_1]\right)$$

$$\leq \frac{\phi(x_1) - \tau f_{\inf}}{\alpha\tau K} + \frac{6h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{1-\beta_1}}{h_{\min}\sqrt{1-\beta_2}} \frac{n}{(1-\frac{\beta_1}{\beta_2})^{3/2}} \left(\frac{1}{K}\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right) - \log(\beta_2)\right)$$
$$+ \left(\frac{\alpha^2}{\tau}G_1(\beta_1,\beta_2) + \frac{\alpha}{\tau}G_2(\beta_1,\beta_2)\right) \frac{n}{(1-\frac{\beta_1}{\beta_2})} \left(\frac{1}{K}\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right) - \log(\beta_2)\right)$$
$$+ \frac{\alpha^2}{\tau}G_3(\beta_1,\beta_2) + \frac{\alpha}{\tau}G_4(\beta_1,\beta_2). \tag{29}$$

Proof. Proof. Consider arbitrary $k \in \mathbb{N}$. Similar to (20), one has that

$$\phi(x_{k+1}) - \phi(x_k) \leq \tau\alpha\nabla f(x_k)^T d_k - \alpha\rho_k\|c_k\|_1 + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\|d_k\|_2^2. \tag{30}$$

Let $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ denote the maximum and minimum eigenvalues, respectively, of a real symmetric matrix argument. By Assumption 3.2 and Lemma 3.2, one finds for any $j \in \{1,\ldots,k-1\}$ that

$$\lambda_{\max}\left(\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{1/2}\right) \leq \sqrt{\epsilon + \sum_{j=1}^{k}\beta_2^{k-j}\left(\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2} - \epsilon\right)} \leq \sqrt{\sum_{j=1}^{k}\beta_2^{k-j}\left(\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2}\right)}$$
$$= \frac{\sqrt{M^2+\kappa_{\nabla f}^2}}{h_{\min}}\sqrt{\frac{1-\beta_2^k}{1-\beta_2}} = \frac{\alpha_k\sqrt{M^2+\kappa_{\nabla f}^2}}{h_{\min}\alpha(1-\beta_1)}.$$

Consequently, one finds that

$$\mathbb{E}[(P_{k-j}\nabla f(x_{k-j}))^T \operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2} P_{k-j}\nabla f(x_{k-j})]$$
$$\geq \mathbb{E}[\lambda_{\min}(\operatorname{diag}(\tilde{s}_{k,j+1}+\epsilon e)^{-1/2})\|P_{k-j}\nabla f(x_{k-j})\|_2^2] \geq \frac{h_{\min}\alpha(1-\beta_1)}{\alpha_k\sqrt{M^2+\kappa_{\nabla f}^2}}\mathbb{E}[\|P_{k-j}\nabla f(x_{k-j})\|_2^2]. \tag{31}$$

Taking total expectation (30), using Lemmas 3.3 and 3.9 and (31), one finds that

$$\begin{aligned}
&\mathbb{E}[\phi(x_{k+1}) - \phi(x_k)] \\
&\leq \tau\alpha\mathbb{E}[\nabla f(x_k)^T d_k] - \alpha\rho_k\mathbb{E}[\|c_k\|_1] + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\mathbb{E}[\|d_k\|_2^2] \\
&\leq \tau\alpha\mathbb{E}[\nabla f(x_k)^T d_k] - \alpha\rho_k\mathbb{E}[\|c_k\|_1] + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)(\mathbb{E}[\|v_k\|_2^2] + \eta_k^2\mathbb{E}[\|t_k\|_2^2]) \\
&\leq -\frac{\tau h_{\min}\alpha(1-\beta_1)}{2h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{j=0}^{k-1}\beta_1^j\mathbb{E}[\|P_{k-j}\nabla f(x_{k-j})\|_2^2] + \tau\alpha\frac{\rho_{\max}\kappa_{\nabla f}\mathbb{E}[\|c_k\|_1]}{\sigma_{\min}} \\
&\quad + \tau\alpha_k\frac{3h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}{h_{\min}\sqrt{1-\beta_1}}\sum_{j=0}^{k-1}\left(\frac{\beta_1}{\beta_2}\right)^j\sqrt{j+1}\mathbb{E}[\|\tilde{t}_{k-j}\|_2^2] \\
&\quad + \tau\alpha_k L_{P\nabla f}^2\alpha^2\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2}\frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\frac{\beta_1(1+\beta_1)}{(1-\beta_1)^3} \\
&\quad + \tau\alpha_k^3 L_{P\nabla f}^2\frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{j=1}^{k-1}\mathbb{E}[\|t_{k-j}\|_2^2]\sum_{l=j}^{k-1}\beta_1^l\sqrt{l} \\
&\quad - \alpha\rho_{\min}\mathbb{E}[\|c_k\|_1] + \tfrac{1}{2}\alpha^2(\tau L_{\nabla f} + L_J)\left(\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2} + \eta_k^2\mathbb{E}[\|t_k\|_2^2]\right).
\end{aligned}$$

Summing over $k \in \{1,\ldots,K\}$ and using the fact that $\{\alpha_k\}$ is nondecreasing, one has

$$\underbrace{\frac{\tau h_{\min}\alpha(1-\beta_1)}{2h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{k=1}^{K}\sum_{j=0}^{k-1}\beta_1^j\mathbb{E}[\|P_{k-j}\nabla f(x_{k-j})\|_2^2]}_{A} + \alpha\rho_{\min}\left(1 - \frac{\tau\kappa_{\nabla f}\rho_{\max}}{\sigma_{\min}\rho_{\min}}\right)\sum_{k=1}^{K}\mathbb{E}[\|c_k\|_1]$$

$$\leq \phi(x_1) - \tau f_{\inf} + \underbrace{\frac{3h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\tau\alpha_K}{h_{\min}\sqrt{1-\beta_1}}\mathbb{E}\left[\sum_{k=1}^{K}\left(\sum_{j=0}^{k-1}\left(\frac{\beta_1}{\beta_2}\right)^j\sqrt{j+1}\|\tilde{t}_{k-j}\|_2^2\right)\right]}_{B}$$
$$+ \underbrace{\frac{\tau\alpha_K^3 L_{P\nabla f}^2\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\mathbb{E}\left[\sum_{k=1}^{K}\left(\sum_{j=1}^{k-1}\|t_{k-j}\|_2^2\sum_{l=j}^{k-1}\beta_1^l\sqrt{l}\right)\right]}_{C} + \frac{\alpha_K^2(\tau L_{\nabla f}+L_J)}{2}\mathbb{E}\left[\sum_{k=1}^{K}\|t_k\|_2^2\right]$$
$$+ K\left(\tau\alpha_K L_{P\nabla f}^2\alpha^2\rho_{\max}^2\kappa_c^2\sigma_{\min}^{-2}\frac{\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\frac{\beta_1(1+\beta_1)}{(1-\beta_1)^3} + \frac{\alpha^2(\tau L_{\nabla f}+L_J)\rho_{\max}^2\kappa_c^2}{2\sigma_{\min}^2}\right). \tag{32}$$

With respect to the term A, one finds by Lemma 3.2 that

$$\begin{aligned}A &= \frac{\tau h_{\min}\alpha(1-\beta_1)}{2h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{k=1}^{K}\sum_{j=0}^{k-1}\beta_1^j\mathbb{E}\left[\|P_{k-j}\nabla f(x_{k-j})\|_2^2\right]\\ &= \frac{\alpha h_{\min}\tau}{2h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{k=1}^{K}(1-\beta_1^{K-k+1})\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2].\end{aligned}$$

With respect to the term B, one finds with Lemma 3.8 and $\beta_1 < \beta_2$ that

$$\begin{aligned}B &= \frac{3h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\tau\alpha_K}{h_{\min}\sqrt{1-\beta_1}}\mathbb{E}\left[\sum_{k=1}^{K}\|\tilde{t}_k\|_2^2\sum_{j=k}^{K}\left(\frac{\beta_1}{\beta_2}\right)^{j-k}\sqrt{1+j-k}\right]\\ &\leq \frac{6h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\tau\alpha_K}{h_{\min}\sqrt{1-\beta_1}}\frac{1}{(1-\frac{\beta_1}{\beta_2})^{3/2}}\mathbb{E}\left[\sum_{k=1}^{K}\|\tilde{t}_k\|_2^2\right].\end{aligned}$$

With respect to the term C, one finds with Lemma 3.8 that

$$C \leq \frac{\tau\alpha_K^3 L_{P\nabla f}^2\sqrt{1-\beta_1}}{4\sqrt{M^2+\kappa_{\nabla f}^2}}\mathbb{E}\left[\sum_{k=1}^{K}\|t_k\|_2^2\sum_{l=0}^{K-1}\beta_1^l\sqrt{l}(l+1)\right] \leq \frac{\tau\alpha_K^3 L_{P\nabla f}^2}{\sqrt{M^2+\kappa_{\nabla f}^2}}\frac{\beta_1}{(1-\beta_1)^2}\mathbb{E}\left[\sum_{k=1}^{K}\|t_k\|_2^2\right].$$

Now it follows from Assumption 3.2 and Lemma 3.2 that $s_{K,i} \leq \frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)}$. Thus, with Lemma 3.7,

$$\begin{aligned}\sum_{k=1}^{K}\|t_k\|_2^2 &= \sum_{i=1}^{n}\sum_{k=1}^{K}t_{k,i}^2 = \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{r_{k,i}^2}{s_{k,i}+\epsilon}\\ &\leq \sum_{i=1}^{n}\frac{1}{(1-\beta_1)(1-\frac{\beta_1}{\beta_2})}\left(\log\left(1+\frac{s_{K,i}}{\epsilon}\right)-K\log(\beta_2)\right)\\ &\leq \frac{n}{(1-\beta_1)(1-\frac{\beta_1}{\beta_2})}\left(\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right)-K\log(\beta_2)\right)\end{aligned}$$

and

$$\sum_{k=1}^{K}\|\tilde{t}_k\|_2^2 = \sum_{i=1}^{n}\sum_{k=1}^{K}\tilde{t}_{k,i}^2 = \sum_{i=1}^{n}\sum_{k=1}^{K}\frac{u_{k,i}^2}{s_{k,i}+\epsilon} \leq \sum_{i=1}^{n}\left(\log\left(1+\frac{s_{K,i}}{\epsilon}\right)-K\log(\beta_2)\right)$$

$$\leq n\left(\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right)-K\log(\beta_2)\right).$$

Hence, with $\alpha_K \leq \alpha\frac{1-\beta_1}{\sqrt{1-\beta_2}}$ and the above bounds for A, B, and C, one has from (32) that

$$\begin{aligned}
&\frac{\alpha\tau h_{\min}}{2h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}}\sum_{k=1}^{K}(1-\beta_1^{K-k+1})\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2]+\alpha\rho_{\min}\left(1-\frac{\tau\kappa_{\nabla f}\rho_{\max}}{\sigma_{\min}\rho_{\min}}\right)\sum_{k=1}^{K}\mathbb{E}[\|c_k\|_1]\\
&\leq \phi(x_1)-\tau f_{\inf}+\frac{6h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\tau\alpha\sqrt{1-\beta_1}}{h_{\min}\sqrt{1-\beta_2}}\frac{n}{(1-\frac{\beta_1}{\beta_2})^{3/2}}\left(\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right)-K\log(\beta_2)\right)\\
&\quad+\left(\alpha^3G_1(\beta_1,\beta_2)+\alpha^2G_2(\beta_1,\beta_2)\right)\frac{n}{(1-\frac{\beta_1}{\beta_2})}\left(\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right)-K\log(\beta_2)\right)\\
&\quad+K\left(\alpha^3G_3(\beta_1,\beta_2)+\alpha^2G_4(\beta_1,\beta_2)\right).
\end{aligned}$$

Diving both sides by $\alpha\tau K$, and using (21) along with $(1-\beta_1^{K-k+1})\geq(1-\beta_1)$, yields the result. □ □

This theorem shows that the long-run average of a positive combination of $\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2]$ and $\mathbb{E}[\|c_k\|_1]$ can be made as small as desired. Consider the right-hand side of (29), which can be written as the sum of

$$\begin{aligned}
A &:= \frac{\phi(x_1)-\tau f_{\inf}}{\alpha\tau K},\\
B &:= \frac{6h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{1-\beta_1}n\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right)}{h_{\min}\sqrt{1-\beta_2}(1-\frac{\beta_1}{\beta_2})^{3/2}K},\\
C &:= \frac{6h_{\max}\sqrt{M^2+\kappa_{\nabla f}^2}\sqrt{1-\beta_1}n\left(-\log(\beta_2)\right)}{h_{\min}\sqrt{1-\beta_2}(1-\frac{\beta_1}{\beta_2})^{3/2}},\\
D &:= \left(\frac{\alpha^2}{\tau}G_1(\beta_1,\beta_2)+\frac{\alpha}{\tau}G_2(\beta_1,\beta_2)\right)\frac{n}{(1-\frac{\beta_1}{\beta_2})}\left(\frac{1}{K}\log\left(1+\frac{M^2+\kappa_{\nabla f}^2}{h_{\min}^2(1-\beta_2)\epsilon}\right)-\log(\beta_2)\right),\\
\text{and}\quad E &:= \frac{\alpha^2}{\tau}G_3(\beta_1,\beta_2)+\frac{\alpha}{\tau}G_4(\beta_1,\beta_2).
\end{aligned}$$

Supposing that $\{\rho_k\}$, $\{h_k\}$, β_1, and ϵ are set and fixed, one can choose the remaining inputs β_2, α, and K to make each of the above terms as small as desired. Specifically, consider first the term C. Observe that $\frac{-\log(\beta_2)}{\sqrt{1-\beta_2}}$ and $(1-\frac{\beta_1}{\beta_2})^{-3/2}$ are both monotonically decreasing in $\beta_2\in(\beta_1,1)$ as $\beta_2\to 1$ and the former decreases to 0 and the latter to $(1-\beta_1)^{-3/2}$, and hence C is monotonically decreasing in β_2 over this range and decreases to 0 as $\beta_2\to 1$. Thus, one can first choose β_2 close enough to 1 such that C is as small as desired, and then consider the value of β_2 as fixed. Next, one can choose $\alpha\in(0,1]$ small enough such that D and E are as small as desired, and then consider the value of α as fixed. Note that D could be further reduced by increasing K. The last parameter to choose is K, where one can choose K large enough such that A and B are as small as desired. In summary, by first choosing β_2 close enough to 1, then choosing α small enough, and finally choosing K large enough, one can make the right-hand side of (29) as small as desired. Consequently, the long-run average of the linear combination of $\mathbb{E}[\|P_k\nabla f(x_k)\|_2^2]$ and $\mathbb{E}[\|c_k\|_1]$ can be made as small as desired.

4 Informed Supervised Machine Learning and Implementation Details

Let us now discuss a methodology for which the algorithms proposed and analyzed in the previous section are particularly well suited. The methodology involves incorporating prior knowledge into a supervised learning process through *hard constraints* that are imposed *during training only*. Both of these highlighted aspects are critical for its effectiveness. The methodology's use of *hard constraints* is in contrast to previously proposed methodologies that incorporate prior knowledge through either (a) soft constraints [37] (i.e., through regularization/penalty terms in the objective function) or (b) designing the prediction function to incorporate knowledge directly [4, 23] (e.g., through neural network layers for which a forward pass requires solving a set of equations or even an optimization problem). By imposing such constraints *during training only*, one can avoid having the trained network require expensive operations for each forward pass. Another key feature of this methodology is that one does not solve the hard-constrained training problem with a penalty-based (e.g., augmented Lagrangian) method. This feature is also critical for the effectiveness of the methodology.

The supervised training of a machine learning model involves solving an optimization problem over a set of parameters of a prediction function, call it $p : \mathbb{R}^{n_f} \times \mathbb{R}^n \to \mathbb{R}^{n_o}$, where n_f is the number of features in an input, n is the dimension of the training/optimization problem, and n_o is the dimension of the output. Denoting known input-output pairs in the form $(a, b) \in \mathbb{R}^{n_f} \times \mathbb{R}^{n_o}$ and given a loss function $\ell : \mathbb{R}^{n_o} \times \mathbb{R}^{n_o} \to \mathbb{R}$, the training/optimization problem can be viewed in expected-loss or empirical-loss minimization form, i.e.,

$$\min_{x \in \mathbb{R}^n} \int_{\mathcal{A} \times \mathcal{B}} \ell(p(a, x), b) \mathrm{d}\mathbb{P}_{A,B}(a, b) \quad \approx \quad \min_{x \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^{N} \ell(p(a_i, x), b_i),$$

where $\mathcal{A}$ is the input domain, $\mathcal{B}$ is the output domain, $\mathbb{P}_{A,B}$ is the input-output probability function, and $\{(a_i, b_i)\}_{i=1}^N \subset \mathbb{R}^{n_f} \times \mathbb{R}^{n_o}$. In our setting, the problem has (hard) constraints on x as well. Generally, these can be formulated in various ways; e.g., expectation, probabilistic, or almost-sure constraints. We contend that for many informed-learning problems—such as for many physics-informed learning problems, as we discuss below—a fixed, small number of constraints suffices to improve training. Given a (small) number m of input-output pairs $\{(a_i^c, b_i^c)\}_{i=1}^m$, the constraints may take the form

$$\phi_i(p(a_i^c, x), b_i^c, \dots) = 0 \quad \text{for all} \quad i \in \{1, \dots, m\},$$

where the arguments to the constraint functions $\{\phi_i\}$ may include additional terms, such as derivatives of the prediction function with respect to inputs and/or model weights; see §5 for specific examples.

Our algorithms from the previous section can be employed in numerous informed-learning contexts (e.g., fair learning [7, 11, 16, 34–36]). For this work, we tested our approach on a few physics-informed learning problems. We emphasize that our goal here is not to test huge-scale, state-of-the-art techniques for physics-informed learning. Rather, we take a few physics-informed learning test problems and train relatively straightforward neural networks in order to demonstrate the relative performance of our proposed algorithms with a soft-constrained approach with Adam scaling [15] and the hard-constrained approach with Adam scaling from [21]. The relative performance of the algorithms would be similar if we were to train much more sophisticated and large-scale neural networks that are being developed in state-of-the-art physics-informed learning. For more on physics-informed learning we direct the reader to, e.g., [6, 14, 19, 26, 29, 32, 33]. The work [5] lies in the physics-informed learning with hard constraints, but is restricted to the hard constraints that the PDE inputs and solutions are linearly related, whereas our method handles general nonlinear constraints. They enforce feasibility via projection, while we allow infeasible iterates, using projection only for momentum. Thus, we do not compare our method with theirs.

Let us now provide an overview of the setting of physics-informed learning that we consider in our experiments. A parametric partial differential equation (PDE) can be written generically as $\mathcal{F}(\phi, u) = 0$, where $(\Phi, \mathcal{U}, \mathcal{V})$ is a triplet of Banach spaces, $\mathcal{F} : \Phi \times \mathcal{U} \to \mathcal{V}$ is a differential operator, $\phi \in \Phi$ represents PDE parameters, and $u \in \mathcal{U}$ denotes a solution of the PDE corresponding to ϕ. The aim is to train a model to learn a mapping from the PDE parameters to a corresponding solution. Let such a mapping be denoted as $\mathcal{G} : \Phi \times \mathbb{R}^d \times \mathbb{R}^n \to \mathcal{U}$, the inputs to which are PDE parameters, a vector encoding information about

the domain of the PDE solution about which one aims to make a prediction (e.g., temporal and/or spatial coordinates), and, say, neural-network model parameters, and the output is a predicted solution value.

For training a model to solve the PDE with potentially no known solution values (see [14]), one can consider a set of training inputs $\{(\phi_i, y_i)\}_{i\in S_1}$ and minimize the average PDE residual over the training inputs. Assuming that, in addition, one has access to observed and/or computed solution data in the form of tuples $\{(\phi_i, y_i, u_i)\}_{i\in S_2}$, one can also aim to minimize the differences between known and predicted solution values. Mathematically, these aims can be expressed as finding x to minimize

$$\frac{1}{|S_1|}\sum_{i\in S_1} \|\mathcal{F}(\phi_i, \mathcal{G}(\phi_i, y_i, x))\|_2^2 \quad \text{and/or} \quad \frac{1}{|S_2|}\sum_{i\in S_2} \|u_i - \mathcal{G}(\phi_i, y_i, x)\|_2^2. \tag{33}$$

Note that the ϕ_i and/or y_i elements in $\{(\phi_i, y_i)\}_{i\in S_1}$ may be the same or different from those in $\{(\phi_i, y_i, u_i)\}_{i\in S_2}$. Additional terms may also be used for training, e.g., pertaining to initial and/or boundary conditions, or pertaining to *partial* physics information. For example, in §5.2, we train a model for which it is known that a mass-balance equation should hold, so our training problem involves residuals for the known mass-balance equation, even though this only defines the physics partially. Overall, if one combines all learning aims into a single objective function—say, with a linear combination involving weights for the different objective terms—then one is employing a *soft-constrained* approach. We contend that a more effective approach can be to take at least a subset of terms and impose them as *hard constraints* during training. For example, with respect to the aims in (33), one might impose constraints such as $\mathcal{F}(\phi_i, \mathcal{G}(\phi_i, y_i, x)) = 0$ for some $i \in S_1$ and/or $u_i = \mathcal{G}(\phi_i, y_i, x)$ for some $i \in S_2$. Our experiments show the benefits of this idea.

5 Numerical Experiments

In this section, we present the results of numerical experiments that compare the performance of our proposed algorithms (`SQP-Heavyball(con)` and `SQP-Adam(con)`, respectively, where ("con" stands for "constrained") versus a soft-constrained approach with Adam scaling (`Adam(unc)` for "unconstrained") [14] and a hard-constrained approach with *projection-less* Adam scaling (`Adam(con)`) [21]. We consider four test problems. A few of them—namely, our 1D spring, 1D Burgers' equation, and 2D Darcy flow problems—have been seen in the literature; see [20, 23]. We also consider a problem from chemical engineering, a modified version of a reaction network proposed in [13]. To ensure a fair comparison, for each test problem, `SQP-Heavyball(con)`, `SQP-Adam(con)`, `Adam(unc)`, and `Adam(con)` use the same objective function, which is usually the data-fitting loss plus PDE residual loss, while the hard-constrained methods `SQP-Heavyball(con)`, `SQP-Adam(con)`, and `Adam(con)` impose additional constraints: the PDE residuals are zero at some input data points. Further details are provided in each problem's subsection. The software uses PyTorch (BSD-3 license). For all experiments, the parameters $\beta = 0.9$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$ were used; see Algorithms 2 and 3. Our numerical experiments were performed using Google CoLaboratory$^{\text{TM}}$ L4 GPU platforms.

5.1 1D Spring

Our first test problem aims to predict the movement of a damped harmonic (mass-spring) oscillator [22] under the influence of a restoring force and friction. For simplicity, our aim was to train a model to predict the movement for known parameters and a single initial condition. (Our later test problems involved more complicated situations; this simple problem and the case of only a single initial condition serves as a good starting point for comparison.) The spring can be described by a linear, homogeneous, second-order ordinary differential equation with constant coefficients, namely, $m\frac{d^2u(t)}{dt^2} + \mu\frac{du(t)}{dt} + ku(t) = 0$ over $t \in [0, 1]$, where we fixed the mass $m = 1$, friction coefficient $\mu = 4$, and spring constant $k = 400$. This corresponds to an under-damped state for which the exact solution with amplitude A and phase ϕ is well known to be $u(t) = e^{-\delta t}(2A\cos(\phi + t\sqrt{w_0^2 - \delta^2}))$, where $\delta = \mu/(2m)$ and $w_0 = \sqrt{k/m}$.

Our aim was to train a neural network with the known ODE and a few observed solution values to be able to predict the height of the spring at any time $t \in [0, 1]$. We used a fully connected neural network

with 1 input neuron (corresponding to t), 3 hidden layers with 32 neurons each, and 1 output neuron (that predicts the spring height at time t). Hyperbolic tangent activation was used at each hidden layer. For the training problems, we used two types of terms: ODE-residual and data-fitting terms. The times at which the ODE-residual terms were defined were 30 evenly spaced points over $[0, 1]$. The times at which the data-fitting terms were defined were 10 evenly spaced points over $[0, 0.4]$. The runs for `Adam(unc)` only considered an objective function where the terms in (33) were combined with a weight of 10^{-4} on the average ODE-residual. The runs for the remaining solvers considered the same objective and included hard constraints for the ODE residual at times $\{\frac{4}{29}, \frac{12}{29}, \frac{21}{29}\}$, i.e., 3 constraints. For all algorithms, we ran a "full-batch" version (i.e., with exact objective gradients employed) and a "mini-batch" version, where in each iteration of the latter version only half of the ODE-residual data points were used. We employed the same two fixed learning rates (i.e., value of α in Algorithms 2 and 3) for each algorithm: 5×10^{-4} and 1×10^{-4}. For the other step-size-related parameters we chose $\rho_k = 1$ and $h_k = 1$ for all $k \in \mathbb{N}$.

Results are provided in Figures 1 and 2. The plots in Figure 1 show that `SQP-Adam(con)` yielded lower objective values (loss) more quickly and achieved better accuracy (i.e., lower mean-squared error) after the training budget expired. They also show that `SQP-Adam(con)` achieved more comparable results for the two learning rates, whereas the other algorithms performed worse for the smaller learning rate. Our results here demonstrate that `SQP-Adam(con)` requires less hyperparameter tuning. The plots in Figure 2 show that the difference in performance can be seen clearly in the predictions that one obtains.

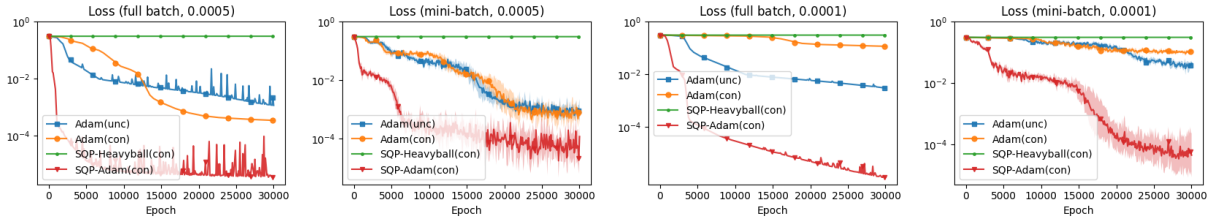

Figure 1: 1D Spring losses over epochs. For the mini-batch runs, the solid lines indicate means over 5 runs while the shaded regions indicate values within one standard deviation of the means.

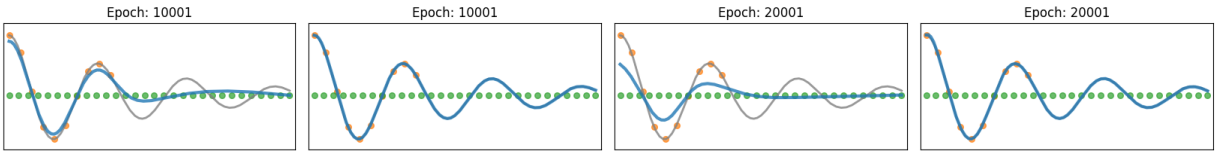

Figure 2: Predicted trajectories. Left to right: `Adam(unc)` (mini-batch, $\alpha_k = 0.0005$), `SQP-Adam(con)` (mini-batch, 0.0005), `Adam(unc)` (mini-batch, 0.0001), and `SQP-Adam(con)` (mini-batch, 0.0001). Axes are time $t \in [0, 1]$ (horizontal) and true/predicted $u(t)$ (vertical). Green dots indicate times at which the ODE-residual terms were defined for the objective function; orange dots indicate data-fitting values; the gray line indicates the true solution; and the blue line indicates the predicted solution. Code from [22] (available under the MIT License) was used to generate the plots.

5.2 Chemical engineering problem

This problem models the reaction system of 1-butene isomerization when cracked on an acidic zeolite [13]. The system is reformulated as an ordinary linear differential equation by scaling the kinetic parameters of

the true model. The ODE is

$$\frac{du(t)}{dt} = \begin{bmatrix} -(c^{(1)} + c^{(2)} + c^{(4)})u^{(1)}(t) + c^{(3)}u^{(3)}(t) + c^{(5)}u^{(4)}(t) \\ 2c^{(1)}u^{(1)}(t) \\ c^{(2)}u^{(1)}(t) - c^{(3)}u^{(3)}(t) \\ c^{(4)}u^{(1)}(t) - c^{(5)}u^{(4)}(t) \end{bmatrix},$$

where $c = [4.283, 1.191, 5.743, 10.219, 1.535]^T$. Our aim was to train a neural network with the known ODE and mass-balance condition (namely, $\frac{du^{(1)}(t)}{dt} + 0.5\frac{du^{(2)}(t)}{dt} + \frac{du^{(3)}(t)}{dt} + \frac{du^{(4)}(t)}{dt} = 0$) over various initial conditions near a nominal initial condition, where the nominal one is $u_0 = [14.5467,\ 16.335,\ 25.947,\ 23.525]$. In this manner, the trained network can be used to predict $u(t)$ at any t (for which we use the range $t \in [0, 10]$) for any initial condition near the nominal one.

We used a fully connected neural network with 5 input neurons (corresponding to the initial condition in $\mathbb{R}^4$ and $t \in \mathbb{R}$), 3 hidden layers with 64 neurons each and hyperbolic tangent activation, and 4 output neurons (corresponding to $u(t) \in \mathbb{R}^4$). The training problems involved three objective terms: ODE-residual (weighted by 10^{-2}), mass-balance (weighted by 10^{-2}), and data-fitting (weighted by 1) terms. Training data was generated by solving the ODE over 1000 initial conditions (of the form $u_0 + \xi$, where ξ was a random vector with each element drawn from a uniform distribution over $[-1, 1]$) using `odeint` from the `scipy` library [31] (BSD licensed). Specifically, solution values were obtained over 64 evenly spaced times in $[0, 10]$, which over the 1000 initial conditions led to 64000 training points. The ODE-residual and mass-balance terms involved all 64000 training points, whereas the data-fitting term involved only 20% of these points chosen at random with equal probability. The runs for `Adam(unc)` used only these objective terms, whereas the runs for `SQP-Heavyball(con)`, `SQP-Adam(con)`, and `Adam(con)` also considered 10 constraints on mass-balance residuals, the points for which were chosen uniformly at random over all initial conditions and times. The mini-batch size was 20% of all samples. We tested learning rates for all algorithms: 5×10^{-4} and 1×10^{-4}. For the other step-size-related parameters we chose $\rho_k = 0.5$ and $h_k = 1$ for all $k \in \mathbb{N}$. The results in Figures 3 show that `SQP-Adam(con)` performed best.

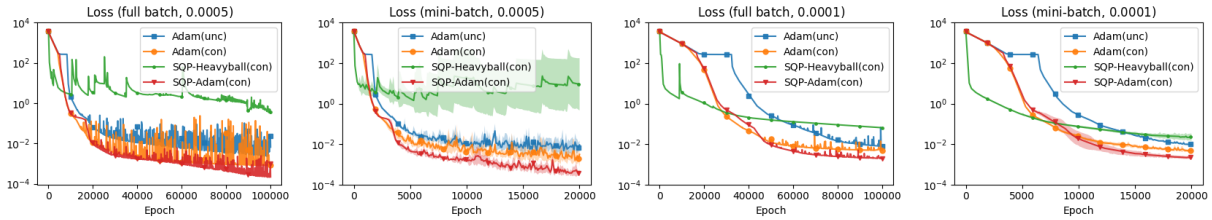

Figure 3: Chemical engineering problem losses over epochs. For the mini-batch runs, solid lines indicate means over 5 runs while the shaded regions indicate values within one standard deviation of the means.

5.3 1D Burgers' equation

Burgers' equation is a PDE often used to describe the behavior of certain types of nonlinear waves [23, 33]. With respect to a spatial domain $[0, 1]$, time domain $[0, 1]$, and viscosity parameter $\nu = 0.01$, we used the equation, initial condition, and (periodic) boundary condition

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} &= \nu \frac{\partial^2 u(x,t)}{\partial x^2}, && x \in (0,1),\ t \in [0,1]; \\ u(x,0) &= u_0(x), && x \in [0,1]; \\ u(x,t) &= u(x+1,t), && x \in [0,1],\ t \in [0,1]. \end{aligned}$$

Our aim was to train a neural network with the known PDE and boundary condition over various initial conditions near a nominal initial condition. In this manner, for any (x, t) and initial condition near the nominal one, the trained network can predict $u(x, t)$.

We used a fully-connected neural network with 34 input neurons (corresponding to x, t, and a discretization of u_0 over 32 evenly spaced points), 3 hidden layers with 64 neurons each and hyperbolic tangent activation, and 1 output neuron (corresponding to $u(x, t)$). The training problems involve three objective terms: PDE-residual (weighted by 10^{-3}), boundary-residual (weighted by 10^{-3}), and data-fitting (weighted by 1) terms. Training data was generated by solving the PDE over 100 initial conditions (of the form $u_0(x) = \sin(2\pi x + \xi\pi)$, where for each instance ξ was chosen from a uniform distribution over $[0, 0.2]$) using the `odeint` solver, as in the previous section. Specifically, solution values were obtained over 32 evenly spaced points each in the spatial and time domains, which over the 100 initial conditions led to 102,400 training points. For each initial condition, the PDE-residual and boundary-residual terms involved all relevant generated training points, whereas the data-fitting term involved only 200 points chosen at random with equal probability. `Adam(unc)` used only these objective terms, whereas `SQP-Heavyball(con)`, `SQP-Adam(con)`, and `Adam(con)` also considered 10 constraints on PDE residuals, the points for which were chosen uniformly at random over all initial conditions and spatio-temporal points. The mini-batch was 20% of all samples. We tested learning rates: 10^{-3} and 5×10^{-4}. For the other step-size-related parameters we chose $\rho_k = 1$ and $h_k = 1$ for all $k \in \mathbb{N}$. One finds in Figure 4 that the results obtained by `Adam(unc)` and `SQP-Adam(con)` were in fact comparable. The performance by the projection-less Adam approach (`Adam(con)`) was inferior to these, and the performance by `SQP-Heavyball(con)` was poorer still. Figure 5 shows that a prediction by the model obtained by `SQP-Adam(con)` is indeed close to the true solution.

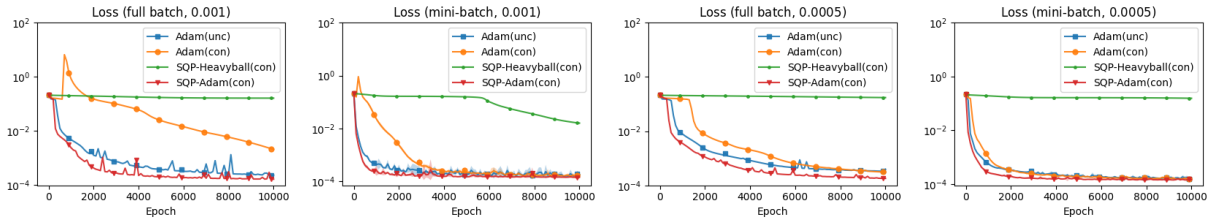

Figure 4: Burgers' losses over epochs. For mini-batch runs, solid lines indicate means over 5 runs while the shaded regions (not very visible) indicate values within one standard deviation of the means.

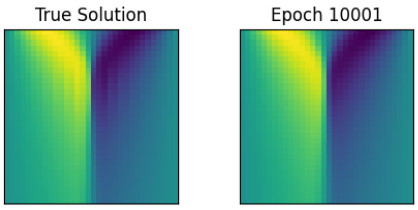

Figure 5: Burgers' true/predicted solutions for initial condition not seen in training. Predicted solution by `SQP-Adam(con)` (mini-batch, $\alpha_k = 0.0005$).

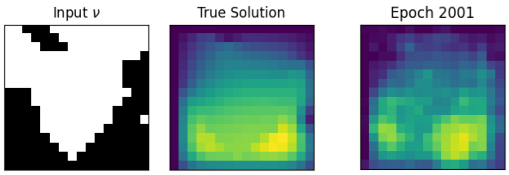

Figure 6: Darcy flow diffusion coefficient ν and true/predicted solution, where diffusion coefficient ν not seen in training. Predicted solution by `SQP-Adam(con)` (mini-batch, $\alpha_k = 0.001$).

5.4 2D Darcy flow

The steady-state 2D Darcy flow equations model the flow of a fluid through a porous medium [23, 29]. With respect to the spatial domain $[0, 1]^2$, a forcing function f (we use $f(x) = 1$ for all $x \in (0, 1)^2$ in our

experiments), and a diffusion coefficient ν, we used

$$\begin{aligned} -\nabla \cdot (\nu(x)\nabla u(x)) &= f(x), && x \in (0,1)^2; \\ u(x) &= 0, && x \in \partial[0,1]^2. \end{aligned}$$

Our aim was to train a neural network with the known PDE and boundary condition over various diffusion coefficients such that, for any $x \in [0,1]^2$ and diffusion coefficient ν, it could be used to predict $u(x)$.

We used the Fourier Neural Operator (FNO) architecture imported from the *neuralop* library [18, 20] (MIT License). The inputs were given in three channels, one for ν, one for a horizontal position embedding, and one for a vertical position embedding. Each channel had dimension 16×16. We used 4 hidden layers (the default). The output was a single channel of dimension 16×16 (corresponding to $u(x)$). The training problems involve three objective terms: PDE-residual (weighted by 10^{-3}), boundary-residual (weighted by 10^{-3}), and data-fitting (weighted by 1) terms. We use the *neuralop* package [17] to generate 100 ν values and their corresponding solutions for training. Specifically, we first generate 1000 samples using the default settings of *neuralop* and then select 100 with similar ν values. For each ν value, the PDE-residual and boundary-residual terms involved all relevant generated training points, whereas the data-fitting term involved only 20% of the points chosen at random with equal probability. The runs for `Adam(unc)` used only these objective terms, whereas the runs for `SQP-Heavyball(con)`, `SQP-Adam(con)`, and `Adam(con)` considered the same objective in addition to 50 constraints on PDE residuals, the points for which were chosen uniformly at random over all initial conditions and spatial points. We ran full-batch and mini-batch settings, where the mini-batch was dictated by 20% of the ν values. We tested using the same learning rates for all algorithms: 5×10^{-3} and 1×10^{-3}. For the other step-size-related parameters we chose $\rho_k = 1$ (`SQP-Heavyball(con)`), $\rho_k = 0.5$ (`SQP-Adam(con)`), and $h_k = 1$ for all $k \in \mathbb{N}$. The results in 7 show strong relative performance by `SQP-Adam(con)`. Figure 6 shows that a prediction by the model obtained by `SQP-Adam(con)` is close to the true solution.

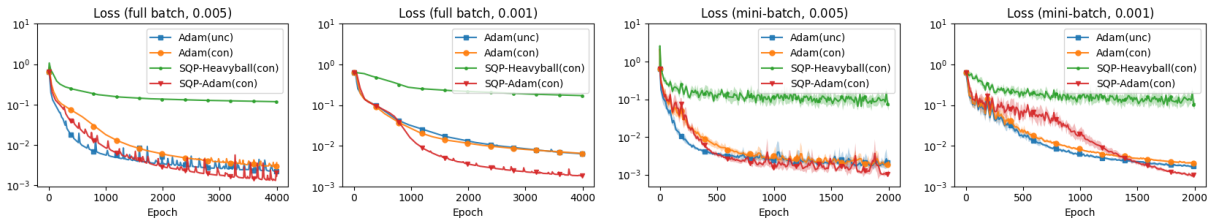

Figure 7: Darcy flow losses over epochs. For mini-batch runs, solid lines indicate means over 5 runs while the shaded regions indicate values within one standard deviation of the means.

6 Conclusion

We proposed two stochastic diagonal-scaling methods for nonlinear equality constrained optimization, and provided convergence guarantees for each approach. We also demonstrated the algorithms in the context of informed supervised learning. The methods' per-iteration costs are comparable to an unconstrained (soft-constrained) approach that also uses diagonal scaling. The numerical experiments reveal practical benefits of the proposed schemes, which we conjecture would also be witnessed when training larger and more sophisticated neural networks for informed learning.

References

[1] Albert S. Berahas, Frank E. Curtis, Michael J. O'Neill, and Daniel P. Robinson. A Stochastic Sequential Quadratic Optimization Algorithm for Nonlinear-Equality-Constrained Optimization with Rank-

Deficient Jacobians. *Mathematics of Operations Research*, page moor.2021.0154, October 2023.

[2] Albert S. Berahas, Frank E. Curtis, Daniel Robinson, and Baoyu Zhou. Sequential Quadratic Optimization for Nonlinear Equality Constrained Stochastic Optimization. *SIAM Journal on Optimization*, 31(2):1352–1379, January 2021.

[3] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, January 2018.

[4] Nithin Chalapathi, Yiheng Du, and Aditi S Krishnapriyan. Scaling physics-informed hard constraints with mixture-of-experts. *arXiv preprint arXiv:2402.13412*, 2024.

[5] Hao Chen, Gonzalo E Constante Flores, and Can Li. Physics-informed neural networks with hard linear equality constraints. *Computers & Chemical Engineering*, 189:108764, 2024.

[6] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88, September 2022.

[7] Frank E. Curtis, Suyun Liu, and Daniel P. Robinson. Fair Machine Learning through Constrained Stochastic Optimization and an ϵ-Constraint Method. *Optimization Letters*, June 2023.

[8] Frank E. Curtis, Michael J. O'Neill, and Daniel P. Robinson. Worst-case complexity of an SQP method for nonlinear equality constrained stochastic optimization. *Mathematical Programming*, June 2023.

[9] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems*. arXiv:1502.04390, 2015.

[10] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A Simple Convergence Proof of Adam and Adagrad. In *Transactions on Machine Learning Research*. arXiv:2003.02395, 2022.

[11] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*. arXiv:1802.08626, 2018.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[13] Udit Gupta, Seongmin Heo, Aditya Bhan, and Prodromos Daoutidis. Time scale decomposition in complex reaction systems: A graph theoretic analysis. *Computers & Chemical Engineering*, 95:170–181, December 2016.

[14] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, May 2021.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*. arXiv:1412.6980, 2015.

[16] Junpei Komiyama, Akiko Takeda, Junya Honda, and Hajime Shimao. Nonconvex optimization for regression with fairness constraints. In *International conference on machine learning*. PMLR, 2018.

[17] Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Valentin Duruisseaux, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. A library for learning neural operators. *arXiv preprint arXiv:2412.10354*, 2025.

[18] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89), 2023.

[19] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[20] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations, ICLR*, 2021.

[21] Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.

[22] Ben Moseley. So, what is a physics-informed neural network? `https://github.com/benmoseley/harmonic-oscillator-pinn/blob/main/Harmonic%20oscillator%20PINN.ipynb`, 2018. Published: 2021-08-28.

[23] Geoffrey Négiar, Michael W. Mahoney, and Aditi S. Krishnapriyan. Learning differentiable solvers for systems with hard constraints. In *International Conference on Learning Representations, ICLR*. arXiv:2207.08675, April 2023.

[24] Michael J. O'Neill. A two stepsize sqp method for nonlinear equality constrained stochastic optimization, 2024.

[25] Boris Teodorovich Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[26] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[27] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[28] Herbert Robbins and David Siegmund. A convergence theorem for nonnegative almost supermartingales and some applications. In Jagdish S. Rustagi, editor, *Optimizing Methods in Statistics*. Academic Press, 1971.

[29] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *Advances in Neural Information Processing Systems*. arXiv:2210.07182, 2023.

[30] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5. RMSPROP: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[31] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[32] Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An Expert's Guide to Training Physics-informed Neural Networks. *arXiv preprint arXiv:2308.08468*, 2023.

[33] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40), October 2021.

[34] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[35] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness Constraints: A Flexible Approach for Fair Classification. *Journal of Machine Learning Research*, 20:1–42, 2019.

[36] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics, AISTATS*, 2017.

[37] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, October 2019.