

An Adaptive Augmented Lagrangian Method for Deterministic and Stochastic Nonconvex Optimization

Tianzhu Liu*

Michael J. O’Neill†

August 27, 2026

Abstract

We present an inexact Augmented Lagrangian algorithm for solving nonlinear, non-convex optimization problems. Unlike most recently proposed Augmented Lagrangian methods with worst-case complexity guarantees, we utilize adaptive penalty parameter updates and full dual stepsizes. We show that the method matches the best known worst-case complexity results for Augmented Lagrangian methods (up to logarithmic factors) when both the function and constraints are deterministic, when the function is stochastic and the constraints are deterministic, and when both are stochastic. Experiments on CUTEst test problems confirm the practical advantages of the proposed approach over Augmented Lagrangian methods with non-adaptive penalty parameters and/or short dual step sizes in the deterministic setting. Numerical results on stochastic constrained optimization problems in machine learning also confirm these findings.

1 Introduction

We consider the following constrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) + g(x) \quad \text{s.t.} \quad c(x) = 0, \quad (1)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuously-differentiable non-convex function, and $g : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ is convex but possibly non-differentiable. In addition, the constraints $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are smooth and continuously differentiable. We also consider cases in which the problem may be stochastic i.e. $f(x) = \mathbb{E}_\xi[F_0(x; \xi)]$ and/or $c(x) = \mathbb{E}_\xi[C(x; \xi)]$.

Applications of deterministic nonlinearly constrained optimization are widespread, including optimal control, PDE-constrained optimization [10], network flow, and resource allocation [7]. In the stochastic setting, applications include physics-informed neural networks [22, 25], Neyman–Pearson classification [19, 30], and many others.

A popular method for solving constrained optimization problems of the form (1) is the Augmented Lagrangian method, which adds a penalty term to the Lagrangian function:

$$\mathcal{L}_\beta(x, y) = f(x) + c(x)^T y + \frac{\beta}{2} \|c(x)\|^2 \quad (2)$$

where $y \in \mathbb{R}^m$ are Lagrange multipliers and $\beta > 0$ is the penalty parameter. Then, a basic Augmented Lagrangian method consists of alternating primal and dual steps of the form:

$$x_{k+1} \in \operatorname{argmin}_{x \in \mathbb{R}^n} \{\mathcal{L}_{\beta_k}(x, y_k) + g(x)\}, \quad y_{k+1} = y_k + \beta_k c(x_{k+1}). \quad (3)$$

*Department of Statistics & Operations Research, UNC–Chapel Hill. Email: tianzhu@unc.edu

†Department of Statistics & Operations Research, UNC–Chapel Hill. Email: mikeoneill@unc.edu

In recent years, many works have proposed Augmented Lagrangian algorithms with worst-case complexity guarantees [3, 11, 19, 20, 29, 33, 34]. For our purposes, we call a primal-dual pair (x, y) an ϵ -approximate stationary solution if

$$\text{dist}(-\nabla_x \mathcal{L}(x, y), \partial g(x)) + \|c(x)\| \leq \epsilon, \quad (4)$$

where $\mathcal{L}(x, y)$ denotes the standard Lagrangian function.

Most Augmented Lagrangian methods with complexity guarantees rely upon two major methodological changes when compared with standard Augmented Lagrangian routines: increasing the penalty parameter at each iteration via a fixed schedule and decoupling the dual stepsize and the penalty parameter. The penalty parameter schedule is chosen to guarantee that no more than $\mathcal{O}(\log(\epsilon^{-1}))$ outer iterations are required to find a point satisfying (4). However, this procedure may increase the penalty parameter far too quickly, causing the subproblems to become highly ill-conditioned and difficult to solve. We empirically validate this deficiency in Section 6.1, where our method which adaptively adjusts the penalty parameter significantly outperforms Augmented Lagrangian methods with fixed schedules.

The second change, that of a decoupled dual stepsize, is arguably the one of much greater theoretical consequence. This change is made exclusively to use a sufficiently short dual stepsize to provide a global upper bound on the norm of the dual variables [29].¹ In contrast, popular Augmented Lagrangian algorithms simply impose a bound on the dual variables by projecting them onto a norm ball of a user-specified radius [1]. We adopt this projection based approach, which is sufficient for deriving complexity guarantees. One driving factor for preferring the penalty parameter as the dual stepsize as opposed to a short dual stepsize comes from the derivation of the Augmented Lagrangian method. It is well known that the Augmented Lagrangian method is equivalent to applying the proximal point method to the dual problem [28]. However, once the penalty parameter is decoupled from the dual stepsize, this equivalence no longer holds. In other words, any Augmented Lagrangian method that uses an alternative dual stepsize is *not proximal point on the dual*.²

Taken together, these changes suggest that most Augmented Lagrangian methods with complexity guarantees are much closer in spirit and theory to pure penalty approaches which suffer from severely ill-conditioned subproblems. Indeed, the standard complexity analysis relies on simply bounding away the contribution of the dual variables and showing convergence via a sufficiently large penalty parameter, exactly as is done in the analysis of pure penalty methods [21, 29]. Some exceptions exist; however, these results rely on (near) initial feasibility conditions [33]. One major exception is the analysis of ALGENCAN, which obtains a complexity result under the condition that the adaptively updated penalty parameter remains bounded [3]. Our proposed algorithm is heavily inspired by ALGENCAN and we extend their analysis to prove that a complexity result always holds; either by their analysis when the penalty parameter remains below a threshold dependent on our desired accuracy or by an analysis akin to [29] when the penalty parameter grows beyond this threshold. In this way, we combine these techniques to prove that our adaptive method obtains the same worst-case complexity as that of the non-adaptive methods, regardless of the behavior of the penalty parameter.

Contributions: We propose an adaptive Augmented Lagrangian method which is closely based on ALGENCAN [1]. In the deterministic setting, under standard smoothness and a mild regularity condition, we establish an iteration bound that matches the best known worst-case complexity (up to logarithmic factors), regardless of the behavior of the penalty parameter. For stochastic objectives and/or stochastic constraints, we combine our adaptive Augmented Lagrangian method with a simple stochastic inner loop and provide complexity results that hold with high probability under natural oracle assumptions. We theoretically prove that the cost of this adaptivity is only logarithmic in the relevant factors and thus our methods are less reliant on accurate parameter settings

¹Some papers [19] allow the upper bound on the norm of the dual variables to increase at a $\mathcal{O}(k)$ rate. Our method and analysis can be easily extended to this case, however we stick with a fixed upper bound for clarity of presentation.

²Of course, the method here only satisfies this interpretation on iterations on which the projection is not triggered.

when compared with prior work. Extensive experiments on PyCUTEst and a Neyman–Pearson classification task corroborate the advantages of the proposed update scheme over short-step and non-adaptive baselines, as well as the value of an adaptive inner loop in stochastic settings.

2 Related Work

One classical approach that can effectively solve problem (1) is the quadratic penalty method [15, 17, 21, 24, 26], which iteratively solves a sequence of unconstrained subproblems involving the original objective plus a quadratic term for the constraint violation, scaled by a chosen penalty parameter. In [17], Lan and Monteiro established a worst-case complexity result of roughly $\tilde{O}(\epsilon^{-1})$ for a first-order quadratic penalty method for solving convex problems. In the nonconvex setting, Kong et al. [15] applied an inexact proximal point method to solve the subproblems and obtained an approximate stationary point in at most $O(\epsilon^{-3})$ gradient evaluations. This result was improved by Lin et al. [21] to $O(\epsilon^{-\frac{5}{2}})$ when the constraints are convex and Slater’s condition holds.

While simple, pure penalty methods commonly suffer from highly ill-conditioned subproblems, as the penalty parameter must become very large to drive the algorithm towards feasibility [26]. The Augmented Lagrangian Method, originally introduced by Hestenes [12] and Powell [27], mitigates this difficulty through the addition of Lagrange multipliers to help drive the method towards feasible solutions. Hong [13] proposed a proximal primal dual algorithm (Prox-PDA) for smooth, nonconvex problems with linear constraints and showed the convergence to first-order stationary points with a sublinear rate. Lan and Monteiro [16] investigated the iteration complexity of a first-order ALM in the convex setting and proved a complexity result of $O(\epsilon^{-1})$ gradient evaluations. Similarly, Xu [34] analyzed the convergence properties of a first-order ALM and proved an improved complexity result of $O(\epsilon^{-\frac{1}{2}}|\log \epsilon|)$ when the problem is strongly convex. The same rate was also obtained by Li et al. [20] when strong convexity holds, and, in the nonconvex setting, a complexity result of $O(\epsilon^{-\frac{5}{2}}|\log \epsilon|)$ was proven under Slater’s condition.

In nonconvex settings, both stochastic and deterministic methods have been studied extensively. Table 1 summarizes a number of relevant works with their assumptions and complexity results. In [3], the authors study the complexity of the ALGENCAN method [1] and demonstrate an outer-iteration worst-case complexity result of $\tilde{O}(\log(\epsilon^{-1}))$ under the assumption that the penalty parameter is bounded above by a constant. Our proposed algorithm is largely inspired by ALGENCAN and we demonstrate that one can obtain an outer iteration complexity of $\tilde{O}(\epsilon^{-1})$ when the penalty parameter does not obey such a bound. Sahin et al. [29] proposed an ALM framework with a short dual stepsize, which guarantees uniformly bounded dual variables. They obtained a complexity result of $\tilde{O}(\epsilon^{-4})$ to first-order stationary points, and $\tilde{O}(\epsilon^{-5})$ to second-order stationary points. Li et al. [18] later improved this result by proposing an inexact proximal-point method (IPPM) and obtained a complexity of $\tilde{O}(\epsilon^{-\frac{5}{2}})$ for nonconvex objectives with linear equality constraints and $\tilde{O}(\epsilon^{-3})$ when the constraints are nonconvex. Similarly to [29], a short dual step size is imposed to bound the dual variables. Xie and Wright [33] analyzed a proximal ALM for nonconvex equality constrained problems and proved a worst-case outer-iteration complexity of $\tilde{O}(\epsilon^{-(2-\eta)})$ for $\eta \in [0, 2]$, where η captures the proximal term and penalty growth trade-off.

When the objective function is stochastic and the constraints are deterministic, Lu et al. [23] proposed a single loop variance reduced scheme and obtained a complexity of $\tilde{O}(\epsilon^{-4})$ under a mean squared smoothness condition. Bollapragada et al. [4] introduced an adaptive sampling technique and showed a sample complexity of $O(\epsilon^{-2})$ for specific choice of penalty parameter and when the objective function is convex and the constraints are linear equalities. Li et al. [19] studied stochastic objective function with stochastic constraints using a momentum-based variance-reduced method and proved a convergence result of $O(\epsilon^{-5})$ for nonconvex functions.

For our proposed method, we can show that under the same assumption, the convergence property depends on the inner-solver and we can still match the best-known complexity result for both

deterministic and stochastic setting.

Method	Objective	Constraint	Assumptions	Complexity
iALM [29]	Deterministic	Deterministic	1, 2	$\tilde{\mathcal{O}}(\epsilon^{-4})$
iPPM [18]	Deterministic	Deterministic	1, 2	$\tilde{\mathcal{O}}(\epsilon^{-3})$
ALGENCAN [3]	Deterministic	Deterministic	1, 2	Outer loop $\mathcal{O}(\log(\epsilon^{-1}))$
Stoc-iALM [19]	Stochastic	Stochastic	1, 2, 3, 4, 5	$\tilde{\mathcal{O}}(\epsilon^{-5})$
Proposed	Deterministic	Deterministic	1, 2	$\tilde{\mathcal{O}}(\epsilon^{-3})$
	Stochastic	Deterministic	1, 2, 3, 4	$\tilde{\mathcal{O}}(\epsilon^{-6})$
	Stochastic	Stochastic	1, 2, 3, 4	$\tilde{\mathcal{O}}(\epsilon^{-7})$
	Stochastic	Deterministic	1, 2, 3, 4, 5	$\tilde{\mathcal{O}}(\epsilon^{-4})$
	Stochastic	Stochastic	1, 2, 3, 4, 5	$\tilde{\mathcal{O}}(\epsilon^{-5})$

Table 1: Comparison of major related methods under different assumptions and settings. Assumptions are listed and discussed further in the following section.

3 Preliminaries

3.1 Notation

We use $\|\cdot\|$ to denote the Euclidean norm. For the convex function $g : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$, the subdifferential set at a point x is denoted by $\partial g(x)$. The distance function from a vector x to a set $\mathcal{X}$ is denoted by $\text{dist}(x, \mathcal{X}) = \min_{z \in \mathcal{X}} \|x - z\|$. For an operator $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $J(x) \in \mathbb{R}^{m \times n}$ denotes the Jacobian of c , where the i^{th} row of $J(x)$ is the vector $\nabla c_i(x) \in \mathbb{R}^n$. For integers $k_1 \leq k_2$, we use the notation $[k_1 : k_2] = \{k_1, \dots, k_2\}$.

3.2 Assumptions

Next, we present our assumptions. The first two assumptions will be used throughout the paper while the latter three are specific to the stochastic setting.

Assumption 1 (Smoothness and Boundedness). *We assume that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$, are smooth and bounded, i.e., there exists $L_f, L_J, L_c \geq 0$ such that*

$$\|\nabla f(x) - \nabla f(v)\| \leq L_f \|x - v\|, \quad \|J(x) - J(v)\| \leq L_J \|x - v\|, \quad \|c(x) - c(v)\| \leq L_c \|x - v\| \quad \forall x, v \in \mathbb{R}^n,$$

and $\lambda'_f, \lambda'_J \geq 0$ such that

$$\|\nabla f(x)\| \leq \lambda'_f, \quad \|J(x)\| \leq \lambda'_J, \quad \forall x \in \mathbb{R}^n.$$

In addition, we assume that $f(x)$ and $g(x)$ are lower bounded by $f_{\min}$ and $g_{\min}$ for all $x \in \mathbb{R}^n$.

This assumption guarantees that the gradient of the objective, the constraints and the Jacobian of the constraints are Lipschitz continuous, which is a very standard assumption in the literature.

Assumption 2 (Regularity). *For all iterates k of our algorithm, we assume that*

$$\nu \|c(x_{k+1})\| \leq \text{dist}(-J(x_{k+1})^T c(x_{k+1}), \frac{\partial g(x_{k+1})}{\beta_k}) \quad (5)$$

for some $\nu > 0$.

The second assumption is also relatively standard, and has been used in a number of recent works on Augmented Lagrangian methods [18, 19, 29]. In addition, in the case where $g(x) = 0$, this condition is equivalent to assuming the Polyak-Lojasiewicz condition on the nonlinear least-squares feasibility problem, $\|c(x)\|^2 = 0$, and always holds when the linearly independent constraint qualification (LICQ) holds at each iteration. Thus, one can view this assumption as a constraint qualification which prevents degeneracy.

Next, we switch to stochastic setting, where we assume access to appropriate stochastic oracles, which we denote with a slight abuse of notation.

Assumption 3 (Stochastic Oracle). *For the stochastic version of problem (1), a stochastic first-order oracle can be accessed. That is, for any $x \in \text{dom}(g)$, the oracle can obtain a sample ξ and return $(\nabla f(x, \xi), c(x, \xi), J(x, \xi))$.*

Now, we consider two standard assumptions on the stochastic oracles.

Assumption 4 (Unbiasedness and Bounded Variance). *For any $x \in \text{dom}(g)$, the stochastic gradient estimates satisfy,*

$$\mathbb{E}_\xi[\nabla f(x, \xi)] = \nabla f(x), \quad \mathbb{E}_\xi[\|\nabla f(x, \xi) - \nabla f(x)\|^2] \leq \sigma_g^2,$$

for some $\sigma_f > 0$. Furthermore, there exists $\sigma_c \geq 0$, such that,

$$\begin{aligned} \mathbb{E}_\xi[c(x, \xi)] &= c(x), \\ \mathbb{E}_\xi[J(x, \xi)] &= J(x), \\ \mathbb{E}_\xi[\|J(x, \xi) - J(x)\|^2] &\leq \sigma_c^2; \\ \mathbb{E}_{\xi_1, \xi_2}[\|J(x, \xi_1)^T c(x, \xi_2) - J(x)^T c(x)\|^2] &\leq \sigma_c^2, \end{aligned}$$

where ξ_1 and ξ_2 are independent and follow the same distribution as ξ .

We note here that we intentionally consider the possibility that $\sigma_c = 0$, which corresponds to the case where the objective is stochastic but the constraints remain deterministic, while not requiring additional analysis specific to this setting.

Finally, we occasionally consider a stronger condition on the stochastic gradient and Jacobian estimates, in which they are sufficiently smooth functions for any fixed sample.

Assumption 5 (Mean-squared Smoothness). *For all $x, v \in \text{dom}(g)$, the stochastic gradient estimates satisfy:*

$$\mathbb{E}_\xi[\|\nabla f(x, \xi) - \nabla f(v, \xi)\|^2] \leq L_f^2 \|x - v\|^2,$$

for some $L_f > 0$. Furthermore,

$$\begin{aligned} \mathbb{E}_\xi[\|J(x, \xi) - J(v, \xi)\|^2] &\leq L_J^2 \|x - v\|^2 \\ \mathbb{E}_{\xi_1, \xi_2}[\|J(x, \xi_1)^T c(x, \xi_2) - J(v, \xi_1)^T c(v, \xi_2)\|^2] &\leq L_J^2 \|x - v\|^2, \end{aligned}$$

where $L_J > 0$ and ξ_1 and ξ_2 are independent and follow the same distribution as ξ .

This last assumption extends smoothness to the stochastic case by bounding the mean-squared variation of the relevant derivatives along sample paths. Notice that this assumption is more restrictive than that of Assumption 4. This assumption is necessary for algorithms based on variance reduction techniques, which in turn achieve stronger complexity guarantees.

4 Algorithm

In this section, we present our proposed adaptive Augmented Lagrangian algorithm as well as a simple inner loop procedure for the stochastic setting.

4.1 Main Algorithm

Algorithm 1 Inexact Augmented Lagrangian Algorithm

Require: $\epsilon \in \mathbb{R}_{>0}$; $\beta_1 \in \mathbb{R}_{>0}$; $x_1 \in \mathbb{R}^n$; $y_1 \in \mathbb{R}^m$; $y_{\max} \in \mathbb{R}_{>0}$; $\gamma > 1$; $\tau \in (0, 1)$; $\{\eta'_k\}_{k \geq 1}$ a decreasing, positive sequence that converging to 0.

1: **for** $k = 1, 2, \dots$ **do**

2: $\eta_k = \min \left\{ \frac{1}{\beta_k}, \eta'_k \right\}$.

3: Find x_{k+1} as an approximate solution to

$$\min_{x \in \mathbb{R}^n} \mathcal{L}_{\beta_k}(x, y_k) + g(x) \quad \text{such that} \quad \text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \eta_k.$$

4: Compute $\hat{y}_{k+1} = y_k + \beta_k c(x_{k+1})$; Quit with x_{k+1} and $\hat{y}_{k+1}$ when

$$\text{dist}(-\nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}), \partial g(x_{k+1})) + \|c(x_{k+1})\| \leq \epsilon;$$

5: If $\|c(x_{k+1})\| \leq \tau \|c(x_k)\|$, then $\beta_{k+1} = \beta_k$; otherwise, $\beta_{k+1} = \gamma \beta_k$.

6: $y_{k+1} = \text{proj}_{[-y_{\max}, y_{\max}]}(\hat{y}_{k+1})$; $k = k + 1$.

7: **end for**

Our main algorithm is presented in Algorithm 1, which follows the general structure of the classical Augmented Lagrangian framework with a few modifications. In general, this structure is largely similar to that of ALGENCAN [3], except that explicit inequality constraints are replaced with the addition of the nonsmooth term g , which allows for the possibility of general inequality constraints via a slack variable reformulation.

At each iteration, we set the tolerance for the inner solver, η_k , as

$$\eta_k = \min \left\{ \frac{1}{\beta_k}, \eta'_k \right\},$$

which incorporates the forcing sequence η'_k , which can be chosen to guarantee that the solution to the inner solver is sufficiently accurate, regardless of the size of β_k . This tolerance is then used for the inexact minimization to find x_{k+1} satisfying

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \eta_k.$$

The dual multiplier is computed in the standard manner,

$$\hat{y}_{k+1} = y_k + \beta_k c(x_{k+1}),$$

with the penalty parameter β_k serving as the dual stepsize. For the purposes of obtaining a complexity result, we also project this update onto the ℓ_∞ norm ball of radius $y_{\max}$.³

The penalty parameter is updated adaptively according to the constraint violation:

$$\beta_{k+1} = \begin{cases} \beta_k, & \text{if } \|c(x_{k+1})\| \leq \tau \|c(x_k)\| \\ \gamma \beta_k, & \text{otherwise} \end{cases}$$

where $0 < \tau < 1$ and $\gamma > 1$. This mechanism increases β_k only when the constraint violation fails to decrease appropriately, preventing over-penalization and maintaining well-conditioned subproblems.

³In the analysis, we rely on a bound on $\|y\|_2$, which implicitly includes a $\sqrt{m}$ factor due to equivalence of norms. This could be easily avoided by projecting onto the ℓ_2 norm ball instead, but we choose the ℓ_∞ ball as it is used in prior work [3].

4.2 Inner Loop Solver for Stochastic Problems

Algorithm 2 Inner Loop Solver using Stochastic Methods

Require: Fixed $y_k \in \mathbb{R}^m$; starting $x_k \in \mathbb{R}^n$; tolerance $\eta_k > 0$; initial epochs $T_{k,1} \in \mathbb{N}_{>0}$; $r > 1$; stochastic proximal method $\mathcal{M}$.

- 1: **for** $p = 1, 2, \dots$ **do**
- 2: Find $x_{k,p}$ as an approximate solution to $\min_x \mathcal{L}_{\beta_k}(x, y_k) + g(x)$ with $T_{k,p}$ iterations of the inner-loop using stochastic method $\mathcal{M}$.
- 3: **if** $\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k,p}, y_k), \partial g(x_{k,p})) > \eta_k$ or $\mathcal{L}_{\beta_k}(x_{k,p}, y_k) + g(x_{k,p}) > \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k)$
- 4: $T_{k,p+1} = rT_{k,p}$.
- 5: **else**
- 6: **Break.**
- 7: **end if**
- 8: **end for**
- 9: Return $x_{k+1} = x_{k,p}$.

Algorithm 2 is designed to approximately minimize the augmented Lagrangian subproblem in the inexact ALM framework when the objective or constraint functions involve stochastic components or large-scale data. Unlike prior work, our proposed algorithm enforces that the approximate solution x_{k+1} satisfies the inexactness condition

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \eta_k,$$

as opposed to simply hoping that the parameters are set such that the stochastic method makes the above result hold in expectation. In practice, it is often difficult to determine the correct number of inner-loop epochs required to achieve this level of accuracy, especially in the presence of stochastic noise. Algorithm 2 provides a simple yet effective adaptive mechanism to address this issue. In Section 6.2, we validate the efficacy of this approach. Additionally, we require that our solution is monotonic with respect to the previous iterate x_k , which enables us to obtain tight bounds on the function value gap at each outer loop iteration, which is critical for our complexity results. This condition allows us to avoid additional unnecessary assumptions, such as boundedness of the domain of g , which has been used in prior work [18, 19], but precludes the possibility of important problem classes, such as inequality constrained problems via a slack variable reformulation.

Algorithm 2 starts from an initial number of epochs $T_{k,0}$ and an initial point x_k . At each iteration p , it applies a stochastic optimization method $\mathcal{M}$ (such as stochastic gradient descent, SVRG, or momentum-based stochastic methods) to perform $T_{k,p}$ epochs of the chosen algorithm to minimize $\mathcal{L}_{\beta_k}(x, y_k) + g(x)$. Here we also assume that we have access to the full batch quantities. If the output is not sufficiently accurate, then the number of epochs is increased by a factor $r > 1$ and the loop is repeated. In Section 5.3, we show that this algorithm terminates, with high probability, in a comparable number of epochs to standard stochastic methods.

5 Convergence Analysis

In this section, we will derive our main results, beginning with a complexity result for Algorithm 1.

5.1 Outer-loop Complexity Result

We start our analysis by establishing an upper bound on the constraint violation across all iterations.

Lemma 5.1. *Let Assumption 1 and Assumption 2 hold and let $\bar{y}_{\max} = \sup\{\|y_k\|_2\}_{k \geq 1}$ (which is guaranteed to exist by the construction of Algorithm 1). Then, we have that:*

$$(i) \quad \|c(x_{k+1})\| \leq \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_k}}{\nu \beta_k} \text{ for all } k \geq 1.$$

$$(ii) \quad \text{There exists } c_{big} := \max\{\|c(x_1)\|, \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_1}}{\nu \beta_1}\} \text{ such that } \|c(x_k)\| \leq c_{big} \text{ for all } k.$$

Proof. From step 3 of Algorithm 1, the stopping criterion for the inner loop subproblem ensures

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \eta_k \leq \frac{1}{\beta_k}.$$

By the definition of the gradient of the Augmented Lagrangian function,

$$\text{dist}(-\nabla f(x_{k+1}) - J(x_{k+1})^T y_k - \beta_k J(x_{k+1})^T c(x_{k+1}), \partial g(x_{k+1})) \leq \frac{1}{\beta_k}.$$

Applying the triangle inequality and rearranging the terms, we have

$$\text{dist}(-\beta_k J(x_{k+1})^T c(x_{k+1}), \partial g(x_{k+1})) \leq \|\nabla f(x_{k+1})\| + \|J(x_{k+1})^T y_k\| + \frac{1}{\beta_k},$$

and therefore,

$$\begin{aligned} \text{dist}(-J(x_{k+1})^T c(x_{k+1}), \frac{\partial g(x_{k+1})}{\beta_k}) &\leq \frac{\|\nabla f(x_{k+1})\| + \|J(x_{k+1})^T y_k\| + \frac{1}{\beta_k}}{\beta_k} \\ &\leq \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_k}}{\beta_k}. \end{aligned}$$

By Assumption 2, we have that $\|c(x_{k+1})\| \leq \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_k}}{\nu \beta_k}$ for all $k \geq 1$. The second result follows by noting that $\beta_k \geq \beta_1$ for all k . $\square$

The uniform bound from the previous lemma allows us to control the constraint violation in terms of the penalty parameter. Using this, we can now derive sufficient conditions on the penalty parameter under which Algorithm 1 is guaranteed to terminate with an approximately feasible solution.

Lemma 5.2. *Let Assumptions 1 and 2 hold. If in any iteration k , we have*

$$\beta_k \geq \beta_{upper} := \frac{\nu + \lambda'_f + \lambda'_J \bar{y}_{\max} + \sqrt{4\epsilon\nu + (\nu + \lambda'_f + \lambda'_J \bar{y}_{\max})^2}}{2\epsilon\nu},$$

then Algorithm 1 terminates at iteration $k+1$ with x_{k+1} and $\hat{y}_{k+1}$ which satisfy the approximate optimality condition (4).

Proof. By step 4 of Algorithm 1, the method terminates when

$$\text{dist}(-\nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}), \partial g(x_{k+1})) + \|c(x_{k+1})\| \leq \epsilon.$$

We consider two terms on the left-hand side of the inequality separately. By Lemma 5.1 part (i), we have that $\|c(x_{k+1})\| \leq \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_k}}{\nu \beta_k}$ holds for all k . Now, by step 3 of Algorithm 1, the stopping criterion for the inner loop subproblem is set as

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \eta_k.$$

Recalling that $\hat{y}_{k+1} = y_k + \beta_k c(x_{k+1})$, we have

$$\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k) = \nabla f(x_{k+1}) + J(x_{k+1})^T \hat{y}_{k+1} = \nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}).$$

Therefore, it follows that

$$\text{dist}(-\nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}), \partial g(x_{k+1})) = \text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) \leq \eta_k \leq \frac{1}{\beta_k}.$$

Now, combining both quantities in the stopping criteria together, we have that

$$\text{dist}(-\nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}), \partial g(x_{k+1})) + \|c(x_{k+1})\| \leq \frac{1}{\beta_k} + \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_k}}{\nu \beta_k}.$$

Thus, Algorithm 1 terminates whenever

$$\frac{1}{\beta_k} + \frac{\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_k}}{\nu \beta_k} \leq \epsilon.$$

Solving this inequality for β_k yields a quadratic equation from which we obtain our desired result. $\square$

The previous two lemmas are mild modifications of Theorem 4.1 of [29], and are sufficient to guarantee convergence when β_k becomes large. We now combine this with a modification of the argument in [3] to derive a complexity result for Algorithm 1.

Theorem 5.3. (*Outer-loop Complexity*) *Let Assumptions 1 and 2 hold. Suppose that there exists $N(\epsilon) \in \{1, 2, \dots\}$ such that for all $k \geq N(\epsilon)$, we have $\eta_k \leq \frac{\epsilon}{2}$. Then,*

(i) *Suppose that $\|c(x_{k+1})\| \leq \tau \|c(x_k)\|$ happens at least*

$$\left\lceil \max \left\{ N(\epsilon), \frac{\log(\frac{2c_{\text{big}}}{\epsilon})}{|\log(\tau)|} \right\} \right\rceil$$

consecutive iterations, then we reach the optimality for c_{big} defined in the Lemma 5.1 part (ii);

(ii) *Otherwise, it takes at most*

$$\left\lceil \max \left\{ N(\epsilon), \frac{\log(\frac{2c_{\text{big}}}{\epsilon})}{|\log(\tau)|} \right\} - 1 \right\rceil \times \left\lceil \frac{\log(\frac{\beta_{\text{upper}}}{\beta_1})}{\log(\gamma)} \right\rceil$$

iterations to reach condition (4) with x_{k+1} , $\hat{y}_{k+1}$, and β_{upper} from Lemma 5.2.

Proof. We first consider case (i). By step 4 of Algorithm 1, the algorithm terminates when

$$\text{dist}(-\nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}), \partial g(x_{k+1})) + \|c(x_{k+1})\| \leq \epsilon.$$

In the proof of Lemma 5.2, we obtain a relationship between the gradient of the Augmented Lagrangian function and the Lagrangian function. Combining the relationship with the assumption on $N(\epsilon)$, we have that

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k+1}, y_k), \partial g(x_{k+1})) = \text{dist}(-\nabla_x \mathcal{L}(x_{k+1}, \hat{y}_{k+1}), \partial g(x_{k+1})) \leq \eta_k \leq \frac{\epsilon}{2}$$

for some x_{k+1} and $\hat{y}_{k+1}$.

Meanwhile, suppose $\|c(x_{k+1})\| \leq \tau \|c(x_k)\|$ happens for at least T consecutive iterations. By Lemma 5.1, we have that $\|c(x_{k+1})\| \leq c_{\text{big}}$ holds for all k .

Thus, the algorithm terminates once when $\max\{N(\epsilon), \tau^T c_{\text{big}}\} \leq \frac{\epsilon}{2}$. Since $0 < \tau < 1$, taking the natural logarithm from the second portion inside the max function yields

$$T \geq \frac{\log(\frac{2c_{\text{big}}}{\epsilon})}{|\log(\tau)|}.$$

We now consider case (ii). In the worst-case scenario, $\|c(x_{k+1})\| \leq \tau \|c(x_k)\|$ happens for exactly

$$\max \left\{ N(\epsilon), \frac{\log(\frac{2c_{\text{big}}}{\epsilon})}{|\log(\tau)|} \right\} - 1 \quad (6)$$

consecutive iterations.

Under such circumstances, it is possible that we fail to reach the optimality from consecutive reduction in constraint violation. However, by Lemma 5.2, once β_k exceeds β_{upper} , the algorithm terminates at $k + 1$. In addition, by step 5 of Algorithm 1, $\beta_{k+1} = \gamma^j \beta_1$, where j is the number of iterations for which $\|c(x_{k+1})\| > \tau \|c(x_k)\|$. Thus, we have that $j \leq \frac{\log(\frac{\beta_{\text{upper}}}{\beta_1})}{\log(\gamma)}$. Combining this with (6) yields the result. $\square$

The previous theorem demonstrates two modes of convergence. First, if the constraint violation decreases sufficiently quickly, then the algorithm terminates without needing to excessively increase the penalty parameter. However, when the constraint violation is not well-behaved, the only option available is to increase penalty parameter significantly, which in turn yields convergence. We note that this behavior is reminiscent to classical convergence results for Augmented Lagrangian methods e.g. [2, Proposition 4.2.3], in which convergence can be driven **either** by good dual variable estimates or a sufficiently large penalty parameter. Most prior work (e.g. [18,19,29]) only focus on a single mode of convergence, that is, when the penalty parameter is sufficiently large. In the worst case, the outer loop requires at most $O(\log^2(\epsilon^{-1}))$ iterations to reach approximate feasibility and stationarity. This logarithmic dependence is a mild additional cost imposed by the adaptive nature of the algorithm.

5.2 Deterministic Complexity Result

In order to prove a complete complexity result, we now consider the cost of solving the Augmented Lagrangian subproblem at each iteration. To this end, we prove the following result about the gradient of the Augmented Lagrangian.

Lemma 5.4. *The gradient of the Augmented Lagrangian function, $\nabla_x \mathcal{L}_{\beta_k}(x, y)$, is Lipschitz continuous in x with parameter $L_k = L_f + \bar{y}_{\text{max}} L_J + \beta_k L_J c_{\text{big}} + \beta_k \lambda'_J L_c$, where, c_{big} and $\bar{y}_{\text{max}}$ are defined in Lemma 5.1.*

Proof. By the triangle inequality, we have

$$\|\nabla \mathcal{L}_{\beta_k}(x, y) - \nabla \mathcal{L}_{\beta_k}(v, y)\| \leq \|\nabla f(x) - \nabla f(v)\| + \bar{y}_{\text{max}} \|J(x) - J(v)\| + \beta_k \|J(x)^T c(x) - J(v)^T c(v)\|.$$

We arrive at the result for the first two terms by applying Assumption 1 directly. For the final term in the last inequality, we use Assumption 1 and the upper bound on $\|c(x)\|$ from Lemma 5.1 to obtain

$$\begin{aligned} \beta_k \|J(x)c(x) - J(v)c(v)\| &\leq \beta_k \|J(x) - J(v)\| \|c(x)\| + \beta_k \|J(v)\| \|c(x) - c(v)\| \\ &\leq (\beta_k L_J c_{\text{big}} + \beta_k \lambda'_J L_c) \|x - v\|, \end{aligned}$$

which proves the result. $\square$

The next lemma plays a key role in connecting proximal gradient updates with the stopping criterion of Algorithm 1. The proof is a minor twist on [32, Lemma 9.5], in the sense that we do not assume convexity of our smooth function.

Lemma 5.5. Fix an outer iteration k . Let the inner subproblem at iteration k be $\phi_k(x) = \mathcal{L}_{\beta_k}(x, y_k) + g(x)$. For any $\alpha_k > 0$, define the proximal operator of the nonsmooth portion $g(x)$ as

$$\text{prox}_{\alpha_k g}(x) = \arg \min_u \left\{ g(u) + \frac{1}{2\alpha_k} \|u - x\|^2 \right\};$$

then for any $\alpha_k > 0$, let the proximal gradient step be

$$\psi_{\alpha_k}(x) = \text{prox}_{\alpha_k g}(x - \alpha_k \nabla_x \mathcal{L}_{\beta_k}(x, y_k));$$

and the proximal gradient mapping as

$$G_{\alpha_k}(x) = \frac{1}{\alpha_k} (x - \psi_{\alpha_k}(x)).$$

Then the following hold:

(i)

$$G_{\alpha_k}(x) - \nabla_x \mathcal{L}_{\beta_k}(x, y_k) \in \partial g(\psi_{\alpha_k}(x)).$$

(ii) For any $z \in \text{dom}(g)$ and $\alpha_k \in (0, \frac{1}{L_k}]$,

$$\phi_k(\psi_{\alpha_k}(x)) \leq \mathcal{L}_{\beta_k}(x, y_k) + g(z) - \langle \nabla_x \mathcal{L}_{\beta_k}(x, y_k), x - z \rangle + \langle G_{\alpha_k}(x), x - z \rangle - \frac{\alpha_k}{2} \|G_{\alpha_k}(x)\|^2.$$

(iii) Moreover, when $\alpha_k = \frac{1}{L_k}$, the stationarity residual at the proximal gradient iterate $\psi_{\alpha_k}(x)$ is bounded by the gradient mapping; namely,

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k), \partial g(\psi_{\alpha_k}(x))) \leq 2\|G_{\alpha_k}(x)\|.$$

Consequently, given the inner subproblem tolerance η_k , if $\|G_{\alpha_k}(x)\| \leq \frac{\eta_k}{2}$, then the stopping criterion of the inner subproblem in Algorithm 1 is satisfied.

Proof. By the first-order optimality condition for the proximal operator, we have

$$\psi_{\alpha_k}(x) = \text{prox}_{\alpha_k g}(x - \alpha_k \nabla_x \mathcal{L}_{\beta_k}(x, y_k)) = \arg \min_u \left\{ g(u) + \frac{1}{2\alpha_k} \|u - (x - \alpha_k \nabla_x \mathcal{L}_{\beta_k}(x, y_k))\|^2 \right\}.$$

Thus, we have

$$0 \in \partial g(\psi_{\alpha_k}(x)) + \frac{1}{\alpha_k} (\psi_{\alpha_k}(x) - (x - \alpha_k \nabla_x \mathcal{L}_{\beta_k}(x, y_k))).$$

Plugging in the definition of $G_{\alpha_k}(x)$ yields part (i).

Next, since $\nabla_x \mathcal{L}_{\beta_k}(x, y_k)$ is L_k -Lipschitz by Lemma 5.4, for $\alpha_k \in (0, \frac{1}{L_k}]$, we have

$$\begin{aligned} \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k) &\leq \mathcal{L}_{\beta_k}(x, y_k) + \langle \nabla_x \mathcal{L}_{\beta_k}(x, y_k), \psi_{\alpha_k}(x) - x \rangle + \frac{L_k}{2} \|\psi_{\alpha_k}(x) - x\|^2 \\ &\leq \mathcal{L}_{\beta_k}(x, y_k) - \alpha_k \langle \nabla_x \mathcal{L}_{\beta_k}(x, y_k), G_{\alpha_k}(x) \rangle + \frac{\alpha_k}{2} \|G_{\alpha_k}(x)\|^2 \end{aligned}$$

Meanwhile, by convexity of g and result of (i), for any $z \in \text{dom}(g)$, we have

$$g(\psi_{\alpha_k}(x)) \leq g(z) + \langle G_{\alpha_k}(x) - \nabla_x \mathcal{L}_{\beta_k}(x, y_k), \psi_{\alpha_k}(x) - z \rangle.$$

Adding the two inequalities and plugging in $\psi_{\alpha_k}(x) = x - \alpha_k G_{\alpha_k}(x)$ yields part (ii).

Finally, we establish the connection between the stopping criterion in Algorithm 1 and the proximal gradient mapping $G_{\alpha_k}(x)$. By part (i), we can choose

$$p := G_{\alpha_k}(x) - \nabla_x \mathcal{L}_{\beta_k}(x, y_k) \in \partial g(\psi_{\alpha_k}(x)),$$

then it follows that

$$\begin{aligned} \text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k), \partial g(\psi_{\alpha_k}(x))) &= \inf_{p \in \partial g(\psi_{\alpha_k}(x))} \| -\nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k) - p \| \\ &\leq \| \nabla_x \mathcal{L}_{\beta_k}(x, y_k) - \nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k) - G_{\alpha_k}(x) \| \\ &\leq \| \nabla_x \mathcal{L}_{\beta_k}(x, y_k) - \nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k) \| + \| G_{\alpha_k}(x) \|. \end{aligned}$$

By Lemma 5.4,

$$\| \nabla_x \mathcal{L}_{\beta_k}(x, y_k) - \nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k) \| \leq L_k \| x - \psi_{\alpha_k}(x) \| = L_k \alpha_k \| G_{\alpha_k}(x) \|.$$

Choose $\alpha_k = \frac{1}{L_k}$, we obtain

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k), \partial g(\psi_{\alpha_k}(x))) \leq (1 + L_k \alpha_k) \| G_{\alpha_k}(x) \| = 2 \| G_{\alpha_k}(x) \|.$$

Therefore, if

$$\| G_{\alpha_k}(x) \| \leq \frac{\eta_k}{2},$$

then

$$\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(\psi_{\alpha_k}(x), y_k), \partial g(\psi_{\alpha_k}(x))) \leq \eta_k.$$

This shows that it suffices to drive the gradient mapping norm below $\frac{\eta_k}{2}$ in order to satisfy the subproblem stopping criterion in Algorithm 1. $\square$

Next, we prove a uniform upper bound on $\mathcal{L}_{\beta_k}(x_k, y_k)$, which will appear in our final complexity result.

Lemma 5.6. *Let Assumptions 1 and 2 hold. In addition, assume that*

$$\mathcal{L}_{\beta_k}(x_{k+1}, y_k) + g(x_{k+1}) \leq \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k)$$

holds at every iteration k . Let

$$\hat{N}(\epsilon) := \left\lceil \max \left\{ N(\epsilon), \frac{\log(\frac{2c_{big}}{\epsilon})}{|\log(\tau)|} \right\} - 1 \right\rceil \quad \text{and} \quad c_{\mathcal{L}} := \frac{(4\nu \bar{y}_{\max} + \gamma(\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_1}))(\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_1})}{2\beta_1 \nu^2 (1 - \gamma^{-1})},$$

where $\bar{y}_{\max}$ is defined in Lemma 5.1. Then, for all k ,

$$\mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k) \leq \mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) + c_{\mathcal{L}} \hat{N}(\epsilon).$$

Proof. By the definition of $\mathcal{L}_{\beta_k}(x_k, y_k)$, the statement of the lemma, and the results of Lemma 5.1,

$$\begin{aligned} \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k) &= \mathcal{L}_{\beta_{k-1}}(x_k, y_{k-1}) + g(x_k) + (y_k - y_{k-1})^T c(x_k) + \frac{\beta_k - \beta_{k-1}}{2} \|c(x_k)\|^2 \\ &\leq \mathcal{L}_{\beta_{k-1}}(x_{k-1}, y_{k-1}) + g(x_{k-1}) + \frac{(4\nu \bar{y}_{\max} + \gamma(\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_1}))(\lambda'_f + \lambda'_J \bar{y}_{\max} + \frac{1}{\beta_1})}{2\nu^2 \beta_{k-1}}. \end{aligned}$$

Applying this bound recursively, we have that

$$\mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k) \leq \mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) + \frac{(4\nu\bar{y}_{\max} + \gamma(\lambda'_f + \lambda'_J\bar{y}_{\max} + \frac{1}{\beta_1}))(\lambda'_f + \lambda'_J\bar{y}_{\max} + \frac{1}{\beta_1})}{2\nu^2} \sum_{j=1}^{k-1} \frac{1}{\beta_{j-1}}.$$

By Theorem 5.3, we know that the maximum number of consecutive iterations before we increase β_k or terminate is bounded by $\hat{N}(\epsilon)$. Thus, for all j , we have that $\beta_j \geq \beta_1 \gamma^{\lfloor k/\hat{N}(\epsilon) \rfloor}$ and thus,

$$\begin{aligned} \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k) &\leq \mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) \\ &\quad + \frac{\hat{N}(\epsilon)(4\nu\bar{y}_{\max} + \gamma(\lambda'_f + \lambda'_J\bar{y}_{\max} + \frac{1}{\beta_1}))(\lambda'_f + \lambda'_J\bar{y}_{\max} + \frac{1}{\beta_1})}{2\beta_1\nu^2} \sum_{j=1}^{\lfloor (k-1)/\hat{N}(\epsilon) \rfloor} \frac{1}{\gamma^j} \\ &\leq \mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) + \frac{\hat{N}(\epsilon)(4\nu\bar{y}_{\max} + \gamma(\lambda'_f + \lambda'_J\bar{y}_{\max} + \frac{1}{\beta_1}))(\lambda'_f + \lambda'_J\bar{y}_{\max} + \frac{1}{\beta_1})}{2\beta_1\nu^2(1 - \gamma^{-1})}, \end{aligned}$$

where the final inequality follows by recognizing the summation as a geometric series with parameter γ^{-1} . $\square$

The mild additional assumption that, the inner subproblem solution is at least as good in function value as the previous iterate can be easily enforced by any monotonic inner solver started at x_k . It should be noted that if the algorithm did not adaptively select β_k and instead updated it at each iteration, then $\hat{N}(\epsilon) = 1$ and the previous result would be independent of ϵ . Thus, the fact that $\mathcal{L}_{\beta_k}(x_k, y_k) = \mathcal{O}(\log(\epsilon^{-1}))$ is a direct consequence of the adaptivity of the algorithm, which we regard as a minor price to pay for the additional benefits of an adaptive method.

Next, we prove that the subproblem is uniformly lower bounded at each k , independently of β_k .

Lemma 5.7. *Let Assumption 1 hold. Then,*

$$\inf_{x \in \text{dom}(g)} \mathcal{L}_{\beta_k}(x, y_k) + g(x) \geq f_{\min} + g_{\min} - \frac{1}{2\beta_1} \bar{y}_{\max}^2.$$

Proof. By the definition of the Augmented Lagrangian,

$$\inf_{x \in \text{dom}(g)} \mathcal{L}_{\beta_k}(x, y_k) + g(x) = \inf_{x \in \text{dom}(g)} f(x) + g(x) + y_k^T c(x) + \frac{\beta_k}{2} \|c(x)\|^2 \geq f_{\min} + g_{\min} - \frac{1}{2\beta_1} \bar{y}_{\max}^2,$$

where we used Young's inequality, the definition of $\bar{y}_{\max}$, and $\beta_k \geq \beta_1$ in the final inequality. $\square$

Given these results, we are prepared to obtain an overall complexity guarantee for the deterministic setting.

Theorem 5.8. *(Overall Complexity for Deterministic Setting)*

Suppose that the conditions of Theorem 5.3 hold. If, at every outer iteration k , the subproblem in Step 3 of Algorithm 1 is solved via the proximal gradient method with $\alpha_k = 1/L_k$, then, the total number of proximal gradient iterations required by Algorithm 1 to return an ϵ -approximate feasible solution is

$$\tilde{\mathcal{O}}(\epsilon^{-3}).$$

Proof. Consider the behavior of the inner subproblem solver for a fixed inner iteration k . By Lemma 5.5(ii) applied to the inner subproblem ϕ_k with $z = x$, for every step of proximal gradient, we have

$$\phi_k(\psi_{\alpha_k}(x)) \leq \phi_k(x) - \frac{\alpha_k}{2} \|G_{\alpha_k}(x)\|^2.$$

We note that this inequality clearly proves monotonicity of the iterates in the inner solver, and thus the results of Lemma 5.6 hold. Summing the above inequality for subproblem iterations $t = 1, \dots, T$, we have

$$\phi_k(x_k^{T+1}) \leq \phi_k(x_k^1) - \frac{\alpha_k}{2} \sum_{t=1}^T \|G_t\|^2,$$

where we use G_t as shorthand to denote $G_{\alpha_k}(x_k^t)$. Rearranging this inequality and using the results of Lemmas 5.6 and 5.7,

$$\frac{\alpha_k}{2} \sum_{t=1}^T \|G_t\|^2 \leq \phi_k(x_k^1) - \phi_k(x_k^{T+1}) \leq \mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) - f_{\min} - g_{\min} + c_{\mathcal{L}} \hat{N}(\epsilon) + \frac{1}{2\beta_1} \bar{y}_{\max}^2,$$

Using $\alpha_k = \frac{1}{L_k}$, we have that

$$\min_{1 \leq t \leq T} \|G_t\|^2 \leq \frac{2L_k(\mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) - f_{\min} - g_{\min} + c_{\mathcal{L}} \hat{N}(\epsilon) + \frac{1}{2\beta_1} \bar{y}_{\max}^2)}{T}.$$

Hence, if

$$T_k \geq \frac{8L_k(\mathcal{L}_{\beta_1}(x_1, y_1) + g(x_1) - f_{\min} - g_{\min} + c_{\mathcal{L}} \hat{N}(\epsilon) + \frac{1}{2\beta_1} \bar{y}_{\max}^2)}{\eta_k^2},$$

then there exists some $t \in \{1, \dots, T_k\}$ such that

$$\|G_t\| \leq \frac{\eta_k}{2}.$$

By part (iii) of Lemma 5.5, the corresponding iterate x_k^{t+1} satisfies

$$\text{dist}(-\nabla_x L_{\beta_k}(x_k^{t+1}, y_k), \partial g(x_k^{t+1})) \leq \eta_k.$$

Therefore, the number of proximal gradient iterations required by the k -th inner solve is

$$T_k = O\left(\frac{L_k \hat{N}(\epsilon)}{\eta_k^2}\right).$$

By Lemma 5.4, we have that $L_k = O(\beta_k)$, while Lemma 5.2 proves that $\beta_k = O(\epsilon^{-1})$. Additionally, by the definition of $\hat{N}(\epsilon)$, $\hat{N}(\epsilon) = O(\log(\epsilon^{-1}))$ while Algorithm 1 sets $\eta_k = \Theta(\epsilon)$. Therefore, each subproblem solve requires at most

$$O\left(\frac{\log(\epsilon^{-1})/\epsilon}{\epsilon^2}\right) = O(\log(\epsilon^{-1})\epsilon^{-3})$$

proximal gradient iterations. Combining this with the logarithmic outer-loop bound from Theorem 5.3 yields the overall complexity up to logarithmic factors. $\square$

5.3 Stochastic Complexity Results

We next turn to the stochastic setting. The first step is to define a stochastic gradient mapping and establish its unbiasedness, smoothness, and variance properties. Meanwhile, we showed the Lipschitz property still holds for the Augmented Lagrangian function under the stochastic setting, provided Assumption 5 is satisfied. These are formalized in the following lemma, which we adopted from [19]. We include the proof for completeness.

Lemma 5.9. *Let Assumptions 1, 3 and 4 hold and fix $y \in \mathbb{R}^m$ such that $\|y\| \leq \bar{y}_{\max}$. Let*

$$\Delta_\beta(x, y, \zeta) = \nabla f(x, \xi_1) + J(x, \xi_1)^T y + \beta J(x, \xi_1)^T c(x, \xi_2),$$

where $\zeta = (\xi_1, \xi_2)$ and ξ_1, ξ_2 are independent and follow the same distribution as ξ . Then for any $x, v \in \text{dom}(g)$, it holds:

$$\mathbb{E}_\zeta[\Delta_\beta(x, y, \zeta)] = \nabla \mathcal{L}_\beta(x, y) \quad \text{and} \quad \mathbb{E}_\zeta[\|\Delta_\beta(x, y, \zeta) - \nabla \mathcal{L}_\beta(x, y)\|^2] \leq \sigma_{\mathcal{L}_\beta}^2,$$

where $\sigma_{\mathcal{L}_\beta} = \sqrt{3\sigma_f^2 + 3\sigma_c^2 \bar{y}_{\max}^2 + 3\beta^2 \sigma_c^2}$. If, in addition, Assumption 5 holds, then,

$$\mathbb{E}_\zeta[\|\Delta_\beta(x, y, \zeta) - \Delta_\beta(v, y, \zeta)\|^2] \leq L_{\mathcal{L}_\beta}^2 \|x - v\|^2,$$

holds for any $x, v \in \text{dom}(g)$, where $L_{\mathcal{L}_\beta} = \sqrt{3L_f^2 + 3L_J^2 \bar{y}_{\max}^2 + 3\beta^2 L_J^2}$.

Proof. Since ξ_1 and ξ_2 are independent, it holds

$$\mathbb{E}_\zeta[\Delta_\beta(x, y, \zeta)] = \mathbb{E}_{\xi_1}[\nabla f(x, \xi_1)] + \mathbb{E}_{\xi_1}[J(x, \xi_1)^T y] + \beta \mathbb{E}_{\xi_1}[J(x, \xi_1)^T] \mathbb{E}_{\xi_2}[c(x, \xi_2)].$$

Since ξ_1 and ξ_2 follow the same distribution of ξ . We have that

$$\mathbb{E}_\zeta[\Delta_\beta(x, y, \zeta)] = \nabla f(x) + J(x)^T y + \beta J(x)^T c(x) = \nabla \mathcal{L}_\beta(x, y).$$

In addition, by the triangle inequality,

$$\begin{aligned} \|\Delta_\beta(x, y, \zeta) - \nabla \mathcal{L}_\beta(x, y)\|^2 &\leq 3\|\nabla f(x, \xi_1) - \nabla f(x)\|^2 \\ &\quad + 3\|J(x, \xi_1) - J(x)\|^2 \|y\|^2 \\ &\quad + 3\beta^2 \|J(x, \xi_1)^T c(x, \xi_2) - J(x)^T c(x)\|^2 \\ &\leq 3(\sigma_f^2 + \sigma_c^2 \bar{y}_{\max}^2 + \beta^2 \sigma_c^2). \end{aligned}$$

Similarly, we can repeat the above proof when Assumption 5 holds to derive the bound for $L_{\mathcal{L}_\beta}$. $\square$

Given the previous lemma, we now provide the following result about the stochastic proximal gradient method. Due to the nature of our inner loop solver, Algorithm 2, we adapt standard results to prove a high probability convergence result, which will be sufficient to show that Algorithm 2 terminates in a reasonable number of iterations with high-probability.

Lemma 5.10. *Let Assumptions 1, 3 and 4 hold and fix outer iteration k for Algorithm 1 and inner iteration p for Algorithm 2. Meanwhile, in Algorithm 2, let the stochastic method $\mathcal{M}$ be the stochastic proximal gradient method applied to $\phi_k(x) := L_{\beta_k}(x, y_k) + g(x)$ for T_p iterations, with stepsize $\alpha_{k,p} > 0$, i.e. at stochastic proximal gradient iteration $t \in \{1, \dots, T_p\}$,*

$$x_{t+1,k,p} = \text{prox}_{\alpha_{k,p}g}(x_{t,k,p} - \alpha_{k,p} \Delta_{\beta_k}(x_{t,k,p}, y_k, \zeta_t)),$$

where $\Delta_\beta(x, y, \zeta)$ is defined in Lemma 5.9. Sample t^* uniformly from $\{1, \dots, T_p\}$ and assume that $\mathcal{L}_{\beta_k}(x, y_k) + g(x)$ is lower bounded for all $x \in \text{dom}(g)$. Let L_k be defined in Lemma 5.4, $\sigma_{\mathcal{L}_{\beta_k}}$ be defined in Lemma 5.9, and let $\Omega_k := \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k) - \min_{x \in \text{dom}(g)} (\mathcal{L}_{\beta_k}(x, y_k) + g(x))$ denote the function value gap. Define the failure event as

$$F_{k,p} := \left\{ \text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k,p}, y_k), \partial g(x_{k,p}))^2 > \eta_k^2 \right\} \cup \left\{ \mathcal{L}_{\beta_k}(x_{k,p}, y_k) + g(x_{k,p}) > \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k) \right\}.$$

Then, when $\alpha_{k,p} = \min \left\{ \frac{1}{2L_k}, \sqrt{\frac{\Omega_k}{L_k \sigma_{\mathcal{L}_{\beta_k}}^2 T_p}} \right\}$ and $x_{k,p} = x_{t^*,k,p}$,

$$\mathbb{P}(F_{k,p}) \leq \frac{72(1 + \frac{1}{\sqrt{2}})^2}{\eta_k^2} \max \left\{ \frac{2L_k \Omega_k}{T_{k,p}}, \sigma_{\mathcal{L}_{\beta_k}} \sqrt{\frac{2L_k \Omega_k}{T_{k,p}}} \right\}.$$

Proof. By Lemma 5.9, the estimator $\Delta_{\beta_k}(x, y_k, \zeta)$ is unbiased for $\nabla_x \mathcal{L}_{\beta_k}(x, y_k)$ and has variance bounded by $\sigma_{\mathcal{L}_{\beta_k}}^2$. Moreover, $\nabla_x \mathcal{L}_{\beta_k}(\cdot, y_k)$ is L_k -Lipschitz by Lemma 5.4 and g is convex, thus ϕ_k is L_k -weakly convex. Hence, the assumptions of [5, Corollary 3.6] apply to $\phi_k(x)$ and thus, $\mathbb{E}[\phi_k(x_{t^*,k,p}) \leq \phi_k(x_{x_k})]$,

$$\mathbb{E}[\|G_{1/L_k}(x_{t^*,k,p})\|^2] \leq 18(1 + \frac{1}{\sqrt{2}})^2 \max \left\{ \frac{2L_k \Omega_k}{T_{k,p}}, \sigma_{\mathcal{L}_{\beta_k}} \sqrt{\frac{2L_k \Omega_k}{T_{k,p}}} \right\},$$

holds, by using [8, Theorem 4.5] to connect the norm of the proximal gradient mapping and the norm of the gradient of the Moreau envelope. We note that the two events in $F_{k,p}^c$ can be simultaneously implied by the same event via a mild modification of the proof of [8, Theorem 4.5], and thus combining this inequality with 5.5 (iii) and Markov's inequality yields the result. $\square$

Given the probabilistic bound in the previous lemma, we are now able to establish worst-case complexity bounds for Algorithm 2 which hold with high probability. In the following, we use the terminology “full oracle evaluations” to denote computation of $\nabla f(x)$ (and possible $c(x)$ and $J(x)$, when the constraints are stochastic) as opposed to a stochastic approximation of these quantities.

Lemma 5.11. (*Inner-loop Complexity for Stochastic Proximal Gradient*) *Let Assumptions 1, 3 and 4 hold and fix outer iteration k for Algorithm 1. Let the conditions of Lemma 5.10 hold. Let $C_k := 72(1 + \frac{1}{\sqrt{2}})^2 \max \left\{ 2L_k \Omega_k, \sigma_{\mathcal{L}_{\beta_k}} \sqrt{2L_k \Omega_k} \right\}$. Then, with probability 1, the inner-loop solver terminates finitely. In addition, with probability $1 - \hat{\delta}$, the total number of evaluations required to satisfy the stopping criteria of Algorithm 2 is no more than*

$$\begin{aligned} & \max \left\{ \left\lceil \frac{r^2 C_k^2 - T_1 \eta_k^4 \hat{\delta}^2}{\eta_k^4 \hat{\delta}^2 (r-1)} \right\rceil, T_1 \right\} && \text{stochastic oracle evaluations,} \\ & \max \left\{ \left\lceil \frac{2}{\log r} \log \left(\frac{C_k}{\eta_k^2 \hat{\delta} \sqrt{T_1}} \right) \right\rceil, 1 \right\} && \text{full oracle evaluations.} \end{aligned}$$

Proof. By the update rule of Algorithm 2, the number of proximal stochastic iterations at iteration p is given by $T_{k,p} = r^p T_1$. By Lemma 5.10, we have

$$\mathbb{P}(F_{k,p}) \leq \frac{C_k}{\eta_k^2 \sqrt{T_{k,p}}} = \frac{C_k}{\eta_k^2 \sqrt{T_1}} r^{-\frac{p}{2}}. \quad (7)$$

Then,

$$\sum_{p=0}^{\infty} \mathbb{P}(F_{k,p}) \leq \frac{C_k}{\eta_k^2 \sqrt{T_1}} \sum_{p=0}^{\infty} r^{-\frac{p}{2}} < \infty,$$

and thus by the Borel-Cantelli Lemma, we have that $\mathbb{P}(F_p \text{ infinitely often}) = 0$. Therefore, the algorithm is guaranteed to converge when $T_{p,k}$ is sufficiently large, which yields the first result.

Next, recalling (7), it follows that the probability of success is at least $1 - \hat{\delta}$ whenever

$$p \geq \left\lceil \frac{2}{\log r} \log \left(\frac{C_k}{\eta_k^2 \hat{\delta} \sqrt{T_1}} \right) \right\rceil := p^*.$$

We note that this is the number of full oracle evaluations, as we require one full evaluation at each iteration of Algorithm 2 to check the stopping criteria. Next, the total number of proximal stochastic gradient iterations needed to satisfy the stopping criteria of Algorithm 2, with probability at least $1 - \hat{\delta}$, is no more than

$$T_1 \sum_{p=0}^{p^*} r^p = T_1 \frac{r^{p^*+1} - 1}{r - 1} \leq \frac{r^2 \frac{C_k^2}{\eta_k^4 \hat{\delta}^2} - T_1}{r - 1} = \frac{r^2 C_k^2 - T_1 \eta_k^4 \hat{\delta}^2}{\eta_k^4 \hat{\delta}^2 (r - 1)}.$$

□

If, in addition to Assumptions 3 and 4, we have the stronger assumption 5, then we can obtain an analogous result when PStorm [19] is used as the stochastic method in Algorithm 2.

Lemma 5.12. (*Inner-loop Complexity for PStorm*) *Let Assumptions 1, 3, 4 and 5 hold and fix outer iteration k for Algorithm 1. Let ϕ_k , Ω_k , and $F_{k,p}$ be defined as in Lemma 5.10, and let $\sigma_{\mathcal{L}_\beta}$ and $L_{\mathcal{L}_\beta}$ be defined as in lemma 5.9. Let the conditions of Lemma 5.10 hold. Let $A_k := 1920^{\frac{3}{2}} 3^{\frac{3}{2}} (\frac{24^2}{10})^{\frac{1}{2}} (\max\{\Omega_k, 1\})$ denotes the numerical constant in Lemma 3 from [19]. Then, with probability 1, the inner-loop solver terminates finitely. In addition, with probability at least $1 - \hat{\delta}$, the total number of evaluations required to satisfy the stopping criteria of Algorithm 2 is no more than*

$$\begin{aligned} & \max \left\{ \left\lceil \frac{r^2 A_k \sigma_{\mathcal{L}_\beta} L_{\mathcal{L}_\beta} - T_1 \eta_k^3 \hat{\delta}^{\frac{3}{2}}}{\eta_k^3 \hat{\delta}^{\frac{3}{2}} (r - 1)} \right\rceil, T_1 \right\} && \text{stochastic oracle evaluations,} \\ & \max \left\{ \left\lceil \frac{1}{\log r} \log \left(\frac{A_k \sigma_{\mathcal{L}_\beta} L_{\mathcal{L}_\beta}}{\eta_k^3 \hat{\delta}^{\frac{3}{2}} T_1} \right) \right\rceil, 1 \right\} && \text{full oracle evaluations.} \end{aligned}$$

Proof. As in the previous Lemma, we take the update rule of Algorithm 2 as $T_{k,p} = r^p T_1$. By Lemma 3 of [19] and the definition of A_k , the output $x_{k,p}$ of the p -th PStorm call satisfies

$$\mathbb{E} \left[\text{dist}(-\nabla_x \mathcal{L}_{\beta_k}(x_{k,p}, y_k), \partial g(x_{k,p}))^2 \right] \leq \left(\frac{A_k \sigma_{\mathcal{L}_{\beta_k}} L_{\mathcal{L}_{\beta_k}}}{T_{k,p}} \right)^{2/3}.$$

In addition, the condition

$$\mathcal{L}_{\beta_k}(x_{k,p}, y_k) + g(x_{k,p}) \leq \mathcal{L}_{\beta_k}(x_k, y_k) + g(x_k)$$

is assumed to hold for every p , so the failure event in $F_{k,p}$ cannot occur. Hence, by Markov's inequality,

$$\mathbb{P}(F_{k,p}) \leq \frac{1}{\eta_k^2} \left(\frac{A_k \sigma_{\mathcal{L}_{\beta_k}} L_{\mathcal{L}_{\beta_k}}}{T_{k,p}} \right)^{\frac{2}{3}} = \left(\frac{A_k \sigma_{\mathcal{L}_{\beta_k}} L_{\mathcal{L}_{\beta_k}}}{\eta_k^3 T_1} \right)^{\frac{2}{3}} r^{-\frac{2p}{3}}.$$

Since $r > 1$, by similar argument as the previous Lemma, we have that $\mathbb{P}(F_p \text{ infinitely often}) = 0$, and the inner-loop solver terminates finitely almost surely.

Next, it is sufficient to choose p so that

$$\left(\frac{A_k \sigma_{\mathcal{L}_{\beta_k}} L_{\mathcal{L}_{\beta_k}}}{\eta_k^3 T_1 r^p} \right)^{2/3} \leq \hat{\delta}.$$

Thus, the number of PStorm calls needed that the probability of success is at least $1 - \hat{\delta}$ whenever

$$p \geq \left\lceil \frac{1}{\log r} \log \left(\frac{A_k \sigma_{\mathcal{L}_{\beta_k}} L_{\mathcal{L}_{\beta_k}}}{\eta_k^3 \hat{\delta}^{\frac{3}{2}} T_1} \right) \right\rceil := p^*$$

Finally, the total number of PStorm iterations needed to satisfy the stopping criteria of Algorithm 2, with probability at least $1 - \hat{\delta}$, is no more than

$$T_1 \sum_{p=0}^{p^*} r^p \leq \frac{r^2 A_k \sigma_{\mathcal{L}_\beta} L_{\mathcal{L}_\beta} - T_1 \eta_k^3 \hat{\delta}^{\frac{3}{2}}}{\eta_k^3 \hat{\delta}^{\frac{3}{2}} (r - 1)}.$$

□

Finally, we combine the outer-loop complexity from Theorem 5.3 with the inner-loop analysis from Lemma 5.11 to obtain the overall complexity of our proposed algorithm in the stochastic setting.

Theorem 5.13. *(Overall Complexity for Stochastic Results using Stochastic Proximal Gradient) Let Assumptions 1, 2, 3, and 4 hold. Let the conditions of Lemma 5.10 hold. Then, if the objective is stochastic and the constraints are deterministic, with probability at least $1 - \delta$, the total number of evaluations required by Algorithm 1 to return an ϵ -approximate feasible solution is*

$$\tilde{\mathcal{O}}(\delta^{-2} \epsilon^{-6}) \text{ stochastic oracle evaluations and } \mathcal{O}(\log(\epsilon^{-1}) \log(\delta^{-1} \epsilon^{-1})) \text{ full oracle evaluations.}$$

If both the objective and the constraints are stochastic, then with probability at least $1 - \delta$, the total number of evaluations required by Algorithm 1 to return an ϵ -approximate feasible solution is

$$\tilde{\mathcal{O}}(\delta^{-2} \epsilon^{-7}) \text{ stochastic oracle evaluations and } \mathcal{O}(\log(\epsilon^{-1}) \log(\delta^{-1} \epsilon^{-1})) \text{ full oracle evaluations.}$$

Proof. Let j denote the number of times Algorithm 1 calls the inner solver, and let $j_{\max}$ be the maximum number invokes the inner solver. By Theorem 5.3, we have $j \leq j_{\max}$ and

$$j_{\max} = \mathcal{O}(\log(\epsilon^{-1})).$$

Now consider one call of the inner solver, corresponding to outer iteration k . By Lemma 5.11, with probability at least $1 - \hat{\delta}$, the inner solver terminates in no more than

$$\left\lceil \frac{r^2 C_k^2 - T_1 \eta_k^4 \hat{\delta}^2}{\eta_k^4 \hat{\delta}^2 (r - 1)} \right\rceil,$$

stochastic oracle evaluations. Since, $\eta_k = \Omega(\epsilon)$, $C_k = \mathcal{O}(\max\{L_k \Omega_k, \sigma_{L_{\beta_k}} \sqrt{L_k \Omega_k}\})$ and $\Omega_k = \mathcal{O}(\log(\epsilon^{-1}))$ by Lemmas 5.6 and 5.7, a single call to the inner solver requires at most

$$\tilde{\mathcal{O}}\left(\frac{\max\{L_k^2, \sigma_{L_{\beta_k}}^2 L_k\}}{\epsilon^4 \hat{\delta}^2}\right),$$

with probability at least $1 - \hat{\delta}$ stochastic oracle evaluations.

To ensure that the probability of failure for the output of Algorithm 1 across all inner solves is at most δ , let $\hat{\delta} := \delta / j_{\max}$. Then, by the union bound,

$$\mathbb{P}(\text{All inner solves terminate quickly}) \geq 1 - j \hat{\delta} \geq 1 - j_{\max} \hat{\delta} = 1 - \delta.$$

Therefore, with probability at least $1 - \delta$, the total number of stochastic evaluations required by Algorithm 1 is bounded by

$$\tilde{\mathcal{O}}\left(\frac{j_{\max}^3 \max\{L_k^2, \sigma_{L_{\beta_k}}^2 L_k\}}{\epsilon^4 \delta^2}\right) = \tilde{\mathcal{O}}\left(\frac{\max\{L_k^2, \sigma_{L_{\beta_k}}^2 L_k\}}{\epsilon^4 \delta^2}\right),$$

Finally, we consider $\max\{L_k^2, \sigma_{L_{\beta_k}}^2 L_k\}$ separately depending on whether the constraints are deterministic or stochastic. When the constraints are deterministic, $\sigma_{L_{\beta_k}}$ term is a constant, so $\max\{L_k^2, \sigma_{L_{\beta_k}}^2 L_k\} = \mathcal{O}(\beta_k^2) = \mathcal{O}(\epsilon^{-2})$, and thus the overall complexity is $\tilde{\mathcal{O}}(\delta^{-2}\epsilon^{-6})$. When the constraints are stochastic, $\sigma_{L_{\beta_k}} = \mathcal{O}(\beta_k) = \mathcal{O}(\epsilon^{-1})$, and therefore the overall complexity is $\tilde{\mathcal{O}}(\delta^{-2}\epsilon^{-7})$.

Finally, by Lemma 5.11 with probability at least $1 - \hat{\delta}$, the number of full oracle evaluations required for a single call of the inner solver is no more than

$$\left\lceil \frac{2}{\log r} \log\left(\frac{C_k}{\eta_k^2 \hat{\delta} \sqrt{T_1}}\right) \right\rceil.$$

Recalling the definition of $\hat{\delta}$ above and $j_{\max}$, it follows that, with probability at least $1 - \delta$, the total number of full oracle evaluations is no more than

$$j_{\max} \left\lceil \frac{2}{\log r} \log\left(\frac{C_k}{\eta_k^2 \hat{\delta} \sqrt{T_1}}\right) \right\rceil = \mathcal{O}(\log(\epsilon^{-1}) \log(\epsilon^{-1} \delta^{-1})),$$

regardless of whether the constraints are deterministic or stochastic. $\square$

Corollary 5.13.1. *(Overall Complexity for Stochastic Results using PStorm)*

Let Assumptions 1, 2, 3, 4, and 5 hold. Let the conditions of Lemma 5.12 hold. Then, if the objective is stochastic and the constraints are deterministic, with probability at least $1 - \delta$, the total number of evaluations required by Algorithm 1 to return an ϵ -approximate feasible solution is

$$\tilde{\mathcal{O}}\left(\delta^{-\frac{3}{2}}\epsilon^{-4}\right) \text{ stochastic oracle evaluations and } \mathcal{O}\left(\log(\epsilon^{-1}) \log(\delta^{-1}\epsilon^{-1})\right) \text{ full oracle evaluations.}$$

If both the objective and the constraints are stochastic, then with probability at least $1 - \delta$, the total number of evaluations required by Algorithm 1 to return an ϵ -approximate feasible solution is

$$\tilde{\mathcal{O}}\left(\delta^{-\frac{3}{2}}\epsilon^{-5}\right) \text{ stochastic oracle evaluations and } \mathcal{O}\left(\log(\epsilon^{-1}) \log(\delta^{-1}\epsilon^{-1})\right) \text{ full oracle evaluations.}$$

6 Numerical Experiments

In this section, we present several numerical results. We evaluate the method in both deterministic and stochastic settings. The deterministic experiments investigate the performance of two choices of the Augmented Lagrangian method: whether the penalty parameter is increased consistently or adaptively, and whether a full or short dual step size is used. The stochastic experiments additionally examine the effectiveness of employing Algorithm 2 to solve the inner subproblems versus a simple, fixed epoch strategy more commonly used in the literature [19].

6.1 Deterministic Experiments

We evaluate the four Augmented Lagrangian variants on constrained optimization problems from the PyCUTEst library [9], which covers both convex and nonconvex objectives with equality and inequality constraints. For a problem, let n and m denote the numbers of variables and constraints, respectively, before introducing slack variables. To keep runtime manageable, we select problems satisfying $n + m < 10^3$. Each one-sided inequality constraint of the form $c_i(x) \leq 0$ is reformulated as $c_i(x) + s_i = 0, s_i \geq 0$, and the nonnegativity restrictions on the slack variables are imposed as bound constraints. In addition, we exclude problems without numerical objective values and certain problems which caused memory issues. All experiments were conducted on a workstation equipped with an Intel Core i9-14900KF processor and 32 GB of RAM. The computations were performed using the CPU only, without GPU acceleration. After applying these criteria, the final test set contains 485 problems.

In the deterministic experiments, we use the L-BFGS-B algorithm as our inner-loop solver, implemented via the SciPy optimization library [31]. For the outer-loop update rules, the four variants form a 2×2 comparison between the penalty parameter update rule and the dual variable step size. The terms *Full* and *Short* refer to the dual variable step size, whereas *Always* and *Adaptive* refer to the penalty parameter update frequency.

For the *Full*-step variants, the multiplier update is performed in the same manner as in Algorithm 1. For the *Short*-step variants, the update is $y_{k+1} = y_k + \alpha_k^{\text{short}} c(x_{k+1})$, with

$$\alpha_k^{\text{short}} = \min \left(\frac{\|c(x_1)\| \log^2 2}{\|c(x_{k+1})\| (k+1) \log^2 (k+2)}, 1 \right),$$

as proposed in [29] and variants of which are used in [18, 19].

The two *Always* variants increase the penalty parameter at every outer iteration according to $\beta_{k+1} = \gamma \beta_k$. On the other hand, the two *Adaptive* variants implement the penalty parameter update as in Algorithm 1.

To be more specific, we list them in order:

1. **Always–Full:** Update the dual variable every iteration using the full step, taking the penalty parameter β as the dual step size; cap dual components by 10^9 for numerical stability.
2. **Always–Short:** Update every iteration using the short step size as in Sahin et al. [29], which bounds the dual variable and yields favorable complexity guarantees.
3. **Adaptive–Full (Algorithm 1):** Use the full step (β_k as dual step size) but update the dual variable only when constraint violation fails to decrease sufficiently.
4. **Adaptive–Short:** As in Always–Short method, but update the dual variable using the adaptive scheme.

We compare the methods using the performance profiles from Dolan and Moré [6]. Let $\mathcal{P}$ denote the set of test problems and let $\mathcal{S}$ denote the set of solver methods. For problem $pr \in \mathcal{P}$ and solver $s \in \mathcal{S}$, let $t_{pr,s}$ be the number of gradient evaluations required to satisfy the termination condition for Algorithm 1. For each problem solved by at least one method, define

$$t_{pr}^* := \min_{s \in \mathcal{S}} t_{pr,s}.$$

The performance ratio is defined as

$$r_{pr,s} := \begin{cases} \frac{t_{pr,s}}{t_{pr}^*}, & \text{if solve } s \text{ solved problem } pr; \\ 100, & \text{otherwise.} \end{cases}$$

Thus, an unsuccessful run receives performance ratio of 100 and does not count as solved for any performance factor. For $\alpha \geq 1$, define the performance profile of solver s as

$$\rho_s(\alpha) := \frac{1}{|\mathcal{P}|} |\{pr \in \mathcal{P} : r_{pr,s} \leq \alpha\}|.$$

The value $\rho_s(1)$ is the fraction of problems for which solver s is the most efficient method, up to ties. Larger values of α measure the robustness of the solver over the test set. We display the profiles for $1 \leq \alpha < 100$, which shows the proportion of problems each solver can solve up to 99 times of the shortest number of gradient evaluations.

Figure 1 reports the performance profiles for four combinations of the termination tolerance and penalty-update parameters. The proposed Adaptive-Full method attains the highest performance profile throughout the displayed range in all four panels, indicating the best overall combination of

efficiency and robustness on this test set. Its advantage becomes more pronounced when penalty increases are triggered more frequently and when the penalty growth factor is larger, suggesting that the method remains effective across a range of parameter choices. The Always-Full method is competitive under the looser termination tolerance, but its performance deteriorates under more aggressive penalty growth. The two short-step variants solve fewer problems within the evaluation budget and are significantly outperformed by the methods using a full dual stepsize across all experiments. Overall, these results suggest that combining a full dual step with an adaptive penalty-update rule provides a better balance between feasibility progress and inner-subproblem conditioning than either increasing the penalty parameter at every iteration or using a short dual step.

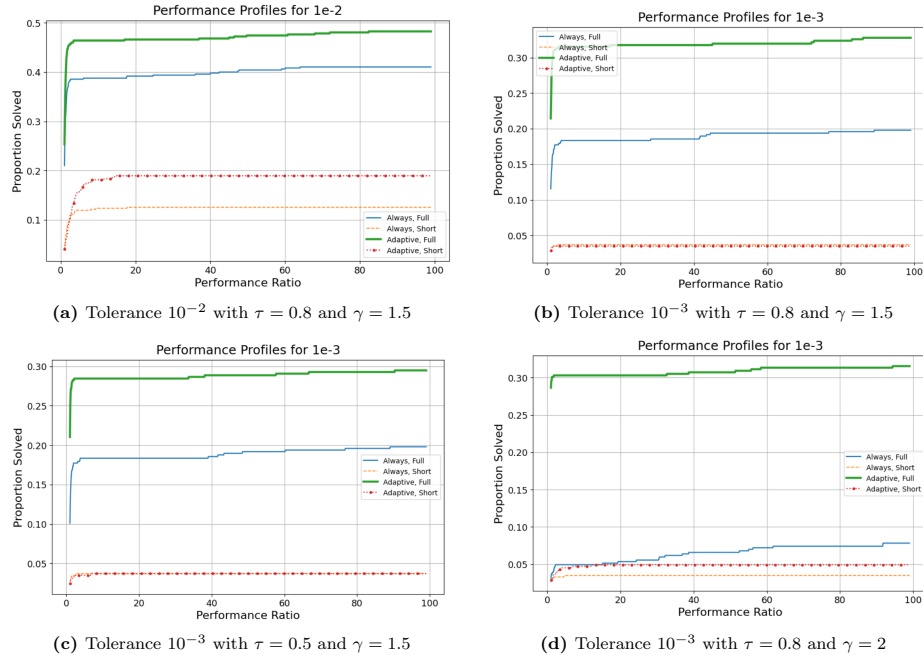


Figure 1: Performance profiles $\rho_s(\alpha)$ for four dual variable update variants. The figures compare the performance profiles of four dual-variable update variants under different stopping tolerances and penalty growth factors. Panels (a)–(b) correspond to two stopping tolerances 10^{-2} , and 10^{-3} respectively. (c) studies a more frequent penalty growth $\tau = 0.5$ with tolerance 10^{-3} . (d) studies a more aggressive penalty growth factor $\gamma = 2$ with tolerance 10^{-3} .

Table 2 below lists a subset of the test problems on which the Always-Full method failed to reach the termination tolerances, while Algorithm 1 (Adaptive-Full) reached an approximate feasible stationary point. From the table, two patterns stand out: (i) the last (before termination) penalty parameter produced by the Always update method is typically several orders of magnitude larger than the penalty parameter under the Adaptive rule (see the column $\beta_{\text{Always}}/\beta_{\text{Adaptive}} \gg 1$), and (ii) the Always-Full method generally accumulates more gradient evaluations yet still fail to converge to the desired accuracy, whereas the Adaptive-Full method succeeds with generally many fewer gradient evaluations. These statistics indicate a systematic difference in how the two methods shape the inner subproblems. This phenomenon is likely due to the conditioning generated by the Augmented Lagrangian subproblem. When the penalty parameter grows aggressively, the curvature introduced by $\frac{\beta}{2} \|c(x)\|^2$ dominates, making the inner problem extremely difficult to solve. The Adaptive update method, by increasing the penalty parameter only when the constraint violation fails to decrease sufficiently, avoids this aggressive penalty parameter growth, which keeps the inner problems reasonably conditioned, and thereby achieves the stopping criteria on these instances.

This evidence supports the incorporation of an adaptive penalty parameter update for practical Augmented Lagrangian methods.

Problem	Always-Full		Algorithm 1		$\beta_{\text{Always}}/\beta_{\text{Adaptive}}$
	Gradient Evaluations	Last β	Gradient Evaluations	Last β	
ALLINITC	1703	3.41376×10^{22}	541	83966.6	4.06562×10^{17}
BROWNALE	1340	2.75095×10^{11}	242	191.751	1.43465×10^9
BT4	1287	4.49548×10^{21}	275	127.834	3.51665×10^{19}
BT6	1265	2.67046×10^{17}	379	55977.7	4.77058×10^{12}
BT7	2503	3.36344×10^{25}	762	7371.55	4.56273×10^{21}
BYRDSPHR	1320	2.99699×10^{21}	148	16.8341	1.78031×10^{20}
CB2	872	2.95282×10^{24}	361	1456.11	2.02788×10^{21}
CB3	881	4.42922×10^{24}	273	287.627	1.53992×10^{22}
CHACONN1	852	2.55411×10^{26}	219	191.751	1.33199×10^{24}
CHACONN2	884	3.36344×10^{25}	244	127.834	2.63110×10^{23}

Table 2: First ten alphabetically ordered test problems with gradient evaluations and final penalty parameters where the Always-Full method fails but Algorithm 1 succeeds with tolerance 10^{-3} , $\tau = 0.8$, and $\gamma = 1.5$.

6.2 Stochastic Experiments

We next evaluate the method in the stochastic setting. Following [19] and [35], we consider the finite sum Neyman-Pearson classification problem on the normalized Spambase dataset [14]. Let $\{a_i^+\}_{i=1}^{n^+}$ and $\{a_i^-\}_{i=1}^{n^-}$ denote the positive and negative class feature vectors, respectively. We solve

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{n^+} \sum_{i=1}^{n^+} \log(1 + \exp(-x^T a_i^+)) \\ \text{s.t.} \quad & \frac{1}{n^-} \sum_{i=1}^{n^-} \log(1 + \exp(x^T a_i^-)) - \hat{c} \leq 0. \end{aligned}$$

The objective function is a logistic surrogate for false-negative error, whereas the constraint is a logistic surrogate to control false-positive error. Here, we consider a range of $\hat{c} \in \{0.6, 0.7, 0.8, 0.9\}$. The dataset is divided into 80% training and 20% test observations using random sampling. Each feature is standardized using the mean and standard deviation computed from the training data and the same transformation is applied to the test set. Our results are averaged over 20 trials, for which the same randomly chosen x_0 is used, while $y_0 = 0$. For shorthand while plotting, we define

$$R = \text{dist}(-\nabla_x L(x, y), \partial g(x)) + \|c(x)\|.$$

We handle the inequality constraint via adding a slack variable s and the constraint $s \geq 0$ and use projected stochastic gradient and PStorm [19] algorithm to solve the stochastic subproblem.

The results in Figure 2 compare the four outer-loop variants when all methods are implemented using Algorithm 2 and projected stochastic gradient is used to solve the subproblem. At each stochastic gradient iteration, positive and negative class mini-batches are sampled randomly with replacement with mini-batch sizes 128, while the stepsize is set to $\frac{1}{\sqrt{T_{k,p}}}$. At the start of the algorithm, we choose $T_{0,0} = 1$ and set $r = 2$. In addition, we set $\beta_0 = 0.1$ and $\gamma = 1.5$, values found after some mild parameter tuning.

Across the values of $\hat{c}$, Adaptive-Full method generally produces the smallest combined residual and the smallest constraint violation. Meanwhile, we notice the Always-Full method initially makes

progress but exhibits larger oscillations and stabilizes at a higher residual level. This difference is especially visible for the looser constraint thresholds. The comparison indicates that the full dual variable step size is more effective when it is paired with adaptive penalty growth. A full dual variable step size alone does not prevent poor numerical behavior when the penalty parameter is increased unnecessarily.

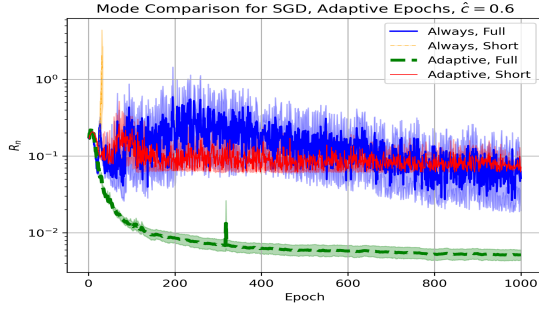
Figure 3 compares methods implementing Algorithm 2 with a fixed epoch strategy for the Always-Full and Adaptive-Full variants. The parameters for the algorithms using Algorithm 2 (labeled “increase” in the plots) are the same as the previous experiment, while the fixed epoch strategies solve each subproblem using 100 epochs with a stepsize of 0.1 and an initial penalty parameter of 1. Across the values of $\hat{c}$, Adaptive-Full method using Algorithm 2 exhibits the most consistent convergence behavior. Overall, the method rapidly decreases to a low residual value and maintains relatively stable values of both the optimality residual and the constraint violation. The advantage is particularly clear for $\hat{c} \in \{0.6, 0.7\}$, where the method reaches lower residuals than its fixed epoch counterpart and displays less variability throughout the run. In general, the increasing epoch strategy provides more stable behavior across all four constraint thresholds and is less sensitive to the choice of a single fixed epoch count. The benefit of increasing inner epoch is especially clear when compared with Always-Full under a fixed epoch, which exhibits worse convergence trajectory for several values of $\hat{c}$, which, while occasionally achieving a lower residual or constraint violation, has consistently erratic behavior across the experiments. These results suggest that progressively increasing the amount of inner-solver computation improves the reliability of the approximate subproblem solutions. The strategy begins with a relatively small computational epoch and increase inner iterations only when the current point fails the acceptance conditions in Algorithm 2. It therefore avoids unnecessarily expensive inner solves during the early iterations while providing greater accuracy when later augmented Lagrangian subproblems become more difficult.

Additional stochastic experiments can be found in Appendix A where PStorm is employed as the inner subproblem solver. The experimental results are qualitatively similar to those for projected stochastic gradient, so we relegate these to the appendix.

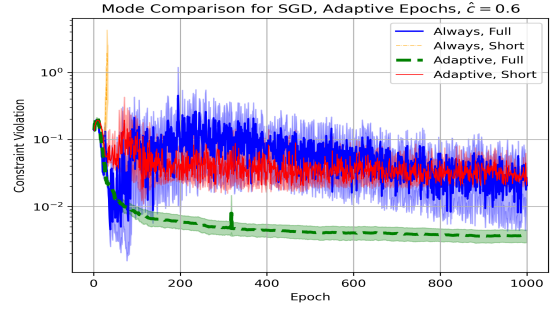
7 Conclusion

In this work, we proposed a simple, adaptive Augmented Lagrangian method for nonlinear, non-convex optimization problems. We prove complexity results that match, up to logarithmic factors, state-of-the-art bounds for Augmented Lagrangian methods under mild conditions. Empirically, we demonstrate that the use of the classical dual stepsize (with a projection step, to enable complexity analysis) as well as an adaptive penalty parameter scheme far outperforms the approaches used by most Augmented Lagrangian methods with complexity guarantees. In addition, we propose a simple, adaptive inner loop for stochastic Augmented Lagrangian methods and derive high probability complexity results when this inner subproblem solver is employed. Numerically, this adaptive inner solver helps stabilize the convergence behavior of the algorithm and proves consistently effective across the set of experiments.

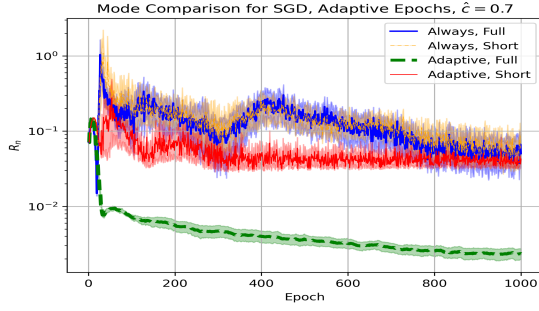
These results apply exclusively to “two-loop” Augmented Lagrangian methods, though there is significant interest in “single-loop” methods, which take simple steps, as opposed to solving a subproblem, at each Augmented Lagrangian iteration. An interesting direction of future research is to explore whether the adaptive strategies proposed here can be extended to single loop Augmented Lagrangian methods.



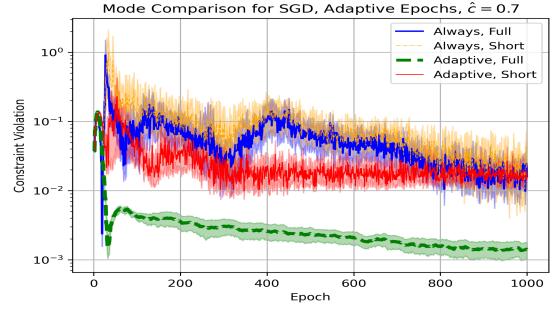
(a) SGD, Adaptive epochs, Accuracy, $\hat{c} = 0.6$.



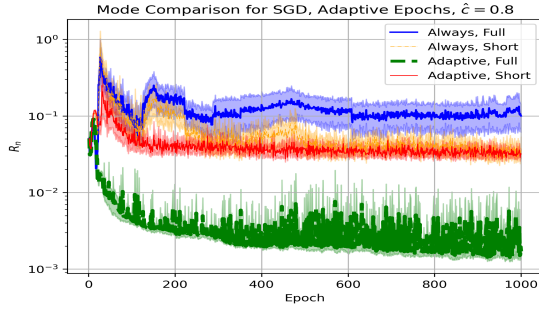
(b) SGD, Adaptive epochs, Constr. violation, $\hat{c} = 0.6$.



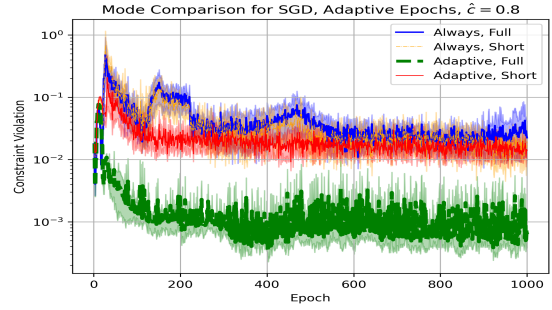
(c) SGD, Adaptive epochs, Accuracy, $\hat{c} = 0.7$.



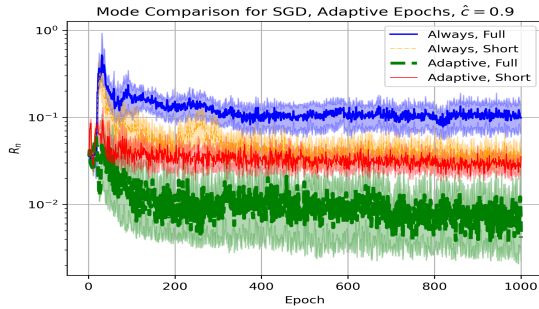
(d) SGD, Adaptive epochs, Constr. violation, $\hat{c} = 0.7$.



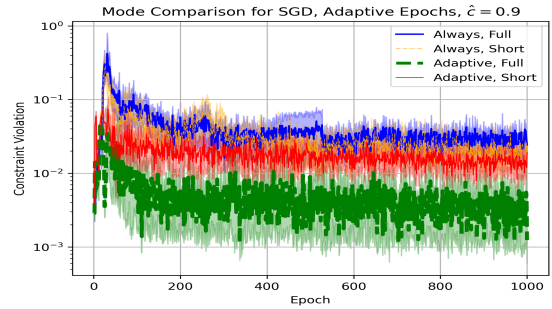
(e) SGD, Adaptive epochs, Accuracy, $\hat{c} = 0.8$.



(f) SGD, Adaptive epochs, Constr. violation, $\hat{c} = 0.8$.

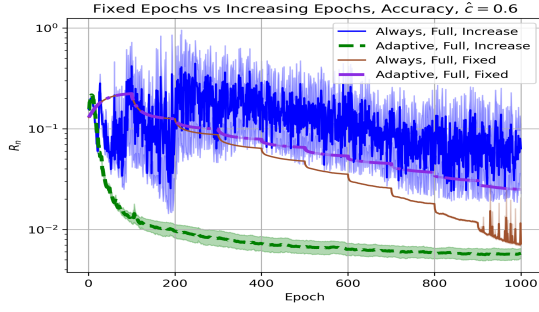


(g) SGD, Adaptive epochs, Accuracy, $\hat{c} = 0.9$.

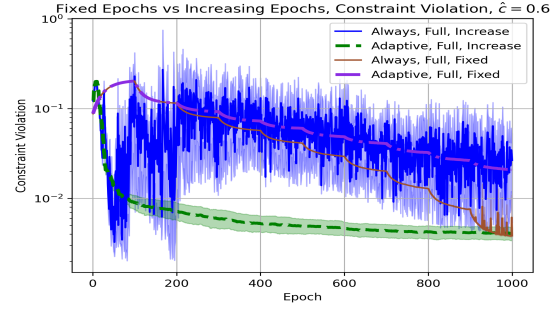


(h) SGD, Adaptive epochs, Constr. violation, $\hat{c} = 0.9$.

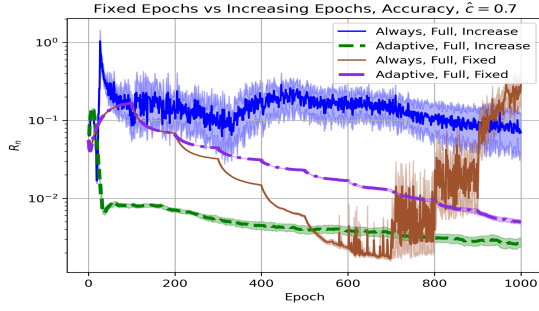
Figure 2: Comparison of different outer loop modes for SGD with adaptive inner epochs varying $\hat{c}$. Average results over 20 trials with 95% confidence intervals.



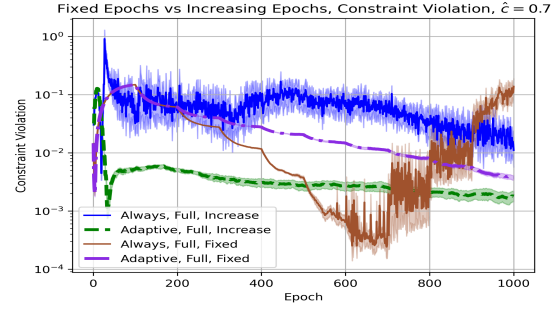
(a) SGD, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.6$.



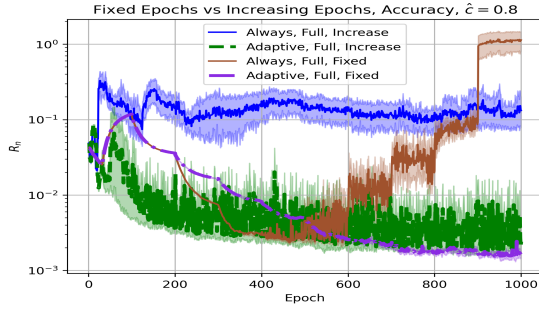
(b) SGD, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.6$.



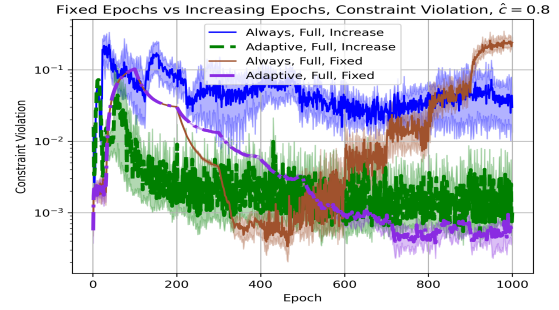
(c) SGD, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.7$.



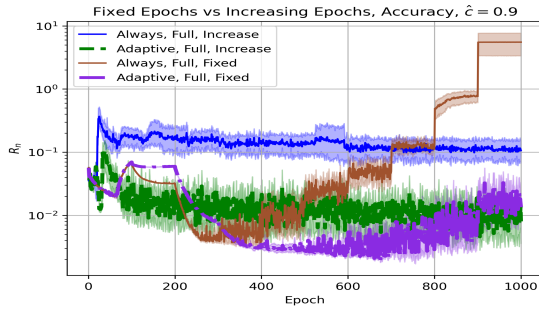
(d) SGD, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.7$.



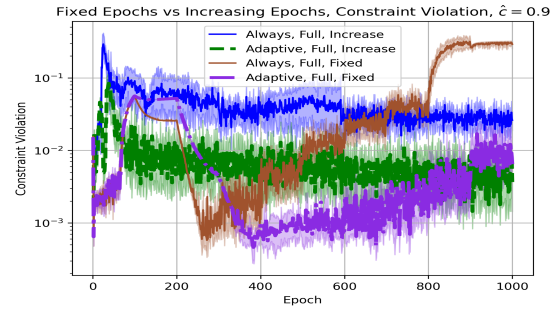
(e) SGD, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.8$.



(f) SGD, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.8$.



(g) SGD, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.9$.



(h) SGD, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.9$.

Figure 3: Comparison of adaptive versus fixed inner epochs for SGD varying $\hat{c}$. Average results over 20 trials with 95% confidence intervals.

References

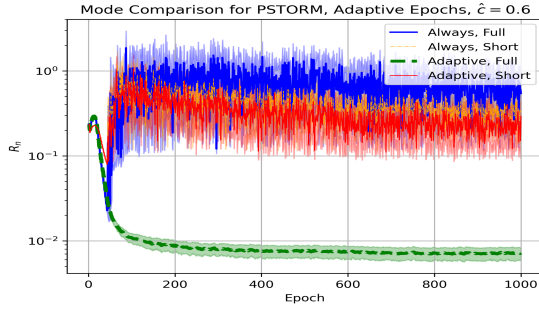
- [1] Roberto Andreani, Ernesto G. Birgin, José Mario Martínez, and María Laura Schuverdt. On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18(4):1286–1309, 2008.
- [2] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999.
- [3] Ernesto G. Birgin and José Mario Martínez. Complexity and performance of an augmented lagrangian algorithm. *Optimization Methods and Software*, 35(5):885–920, 2020.
- [4] Raghu Bollapragada, Cem Karamanli, Brendan Keith, Boyan Lazarov, Socratis Petrides, and Jingyi Wang. An adaptive sampling augmented lagrangian method for stochastic optimization with deterministic constraints. *Computers & Mathematics with Applications*, 149:239–258, 2023.
- [5] Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.
- [6] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.
- [7] Mohammadreza Doostmohammadian, Alireza Aghasi, Maria Vrakopoulou, and Themistoklis Charalambous. 1st-order dynamics on nonlinear agents for resource allocation over uniformly-connected networks. In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1184–1189. IEEE, 2022.
- [8] Dmitriy Drusvyatskiy and Adrian S Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of operations research*, 43(3):919–948, 2018.
- [9] Jaroslav Fowkes, Lindon Roberts, and Árpád Bűrmen. Pycutest: an open source python package of optimization test problems. *Journal of Open Source Software*, 7(78):4377, 2022.
- [10] Fabian Gnengel, Armin Fügenschuh, Michael Hagel, Sven Leyffer, and Marcus Stiemer. A solution framework for linear pde-constrained mixed-integer problems. *Mathematical Programming*, 188(2):695–728, 2021.
- [11] Geovani Nunes Grapiglia and Ya-xiang Yuan. On the complexity of an augmented lagrangian method for nonconvex optimization. *IMA Journal of Numerical Analysis*, 41(2):1546–1568, 2021.
- [12] M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.
- [13] Mingyi Hong. Decomposing linearly constrained nonconvex problems by a proximal primal dual approach: Algorithms, convergence, and applications. *arXiv preprint arXiv:1604.00543*, 2016.
- [14] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository. <https://archive.ics.uci.edu>.
- [15] Weiwei Kong, Jefferson G Melo, and Renato DC Monteiro. Complexity of a quadratic penalty accelerated inexact proximal point method for solving linearly constrained nonconvex composite programs. *SIAM Journal on Optimization*, 29(4):2566–2593, 2019.
- [16] G. Lan and R. D. Monteiro. Iteration-complexity of first-order augmented lagrangian methods for convex programming. *Mathematical Programming*, 155(1-2):511–547, 2016.

- [17] Guanghui Lan and Renato DC Monteiro. Iteration-complexity of first-order penalty methods for convex programming. *Mathematical Programming*, 138(1):115–139, 2013.
- [18] Z. Li, P.-Y. Chen, S. Liu, S. Lu, and Y. Xu. Rate-improved inexact augmented lagrangian method for constrained nonconvex optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2170–2178. PMLR, 2021.
- [19] Z. Li, P.-Y. Chen, S. Liu, S. Lu, and Y. Xu. Stochastic inexact augmented lagrangian method for nonconvex expectation constrained optimization. *Computational Optimization and Applications*, pages 1–31, 2023.
- [20] Z. Li and Y. Xu. Augmented lagrangian based first-order methods for convex and nonconvex programs: Nonergodic convergence and iteration complexity. *arXiv preprint arXiv:2003.08880v1*, 2020.
- [21] Qihang Lin, Runchao Ma, and Yangyang Xu. Complexity of an inexact proximal-point penalty method for constrained smooth non-convex optimization. *Computational optimization and applications*, 82(1):175–224, 2022.
- [22] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.
- [23] Zhaosong Lu, Sanyou Mei, and Yifeng Xiao. Variance-reduced first-order methods for deterministically constrained stochastic nonconvex optimization with strong convergence guarantees. *arXiv preprint arXiv:2409.09906*, 2024.
- [24] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.
- [25] Saviz Mowlavi and Saleh Nabi. Optimal control of pdes using physics-informed neural networks. *Journal of Computational Physics*, 473:111731, 2023.
- [26] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.
- [27] M. J. Powell. A method for nonlinear constraints in minimization problems. In *Optimization*, pages 283–298. 1969.
- [28] R Tyrrell Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of operations research*, 1(2):97–116, 1976.
- [29] M. F. Sahin, A. Alacaoglu, F. Latorre, and V. Cevher. An inexact augmented lagrangian framework for nonconvex optimization with nonlinear constraints. *Advances in Neural Information Processing Systems*, 32, 2019.
- [30] Xin Tong, Yang Feng, and Anqi Zhao. A survey on neyman-pearson classification and suggestions for future research. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(2):64–81, 2016.
- [31] P. Virtanen et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.
- [32] Stephen J Wright and Benjamin Recht. *Optimization for data analysis*. Cambridge University Press, 2022.
- [33] Yue Xie and Stephen J. Wright. Complexity of proximal augmented lagrangian for nonconvex optimization with nonlinear equality constraints. *arXiv preprint arXiv:1908.00131*, 2019.

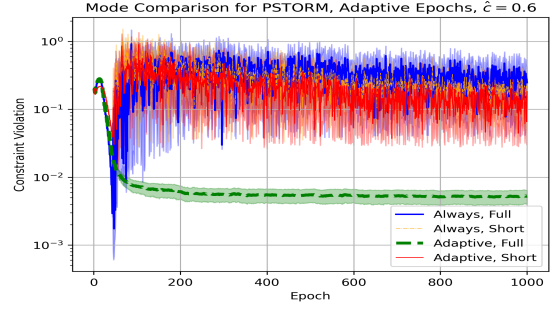
- [34] Y. Xu. Iteration complexity of inexact augmented lagrangian methods for constrained convex programming. *Mathematical Programming*, 185:199–244, 2021.
- [35] Yonggui Yan and Yangyang Xu. Adaptive primal-dual stochastic gradient method for expectation-constrained convex stochastic programs. *Mathematical Programming Computation*, 14(2):319–363, 2022.

A PStorm Experiments

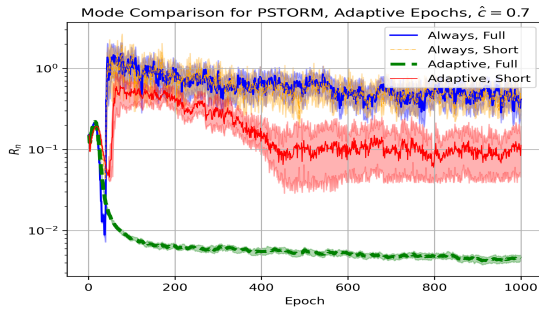
In this section, we repeat the stochastic experiments of Section 6.2 using the PSTORM [19] subproblem solver, which incorporates recursive variance reduction. The results of these experiments can be found in Figures 4 and 5. As noted in Section 6.2, the results are qualitatively similar to those performed using projected stochastic gradient. In particular, the Adaptive, Full method consistently outperforms the other methods tested, across the different problems tested. In addition, the adaptive inner-solver is significantly more stable than the other methods tested and performs as well or better than the non-adaptive methods in most trials.



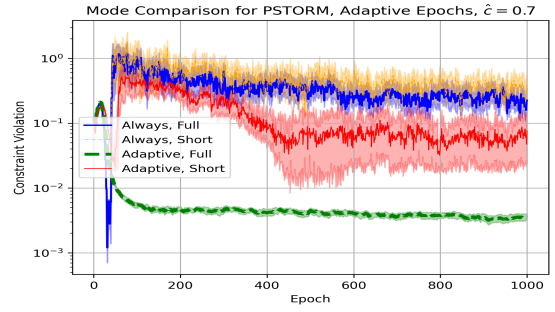
(a) PStorm, Adaptive epochs, Accuracy, $\hat{c} = 0.6$.



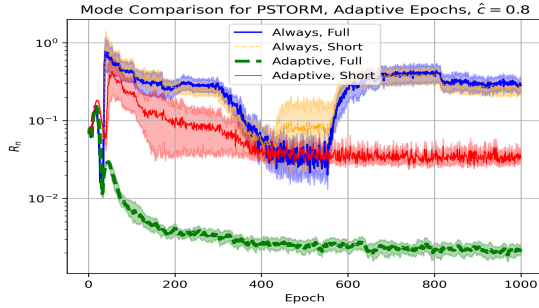
(b) PStorm, Adaptive epochs, Constr. violation, $\hat{c} = 0.6$.



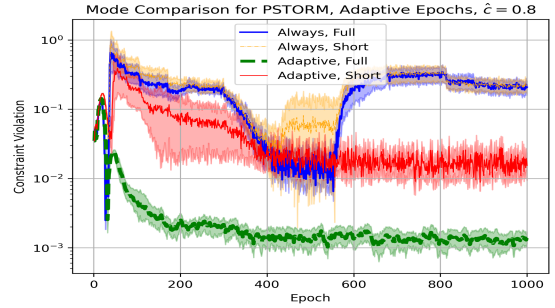
(c) PStorm, Adaptive epochs, Accuracy, $\hat{c} = 0.7$.



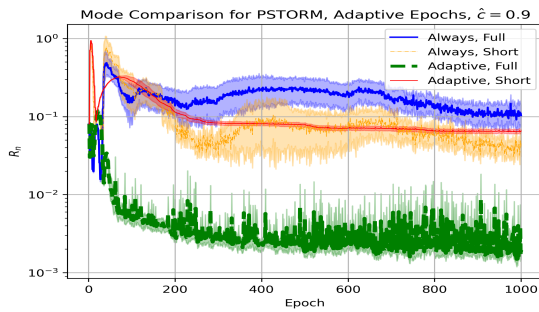
(d) PStorm, Adaptive epochs, Constr. violation, $\hat{c} = 0.7$.



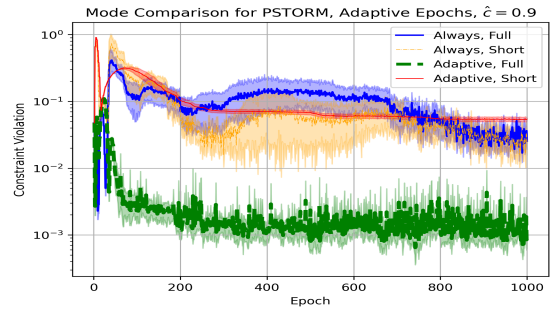
(e) PStorm, Adaptive epochs, Accuracy, $\hat{c} = 0.8$.



(f) PStorm, Adaptive epochs, Constr. violation, $\hat{c} = 0.8$.

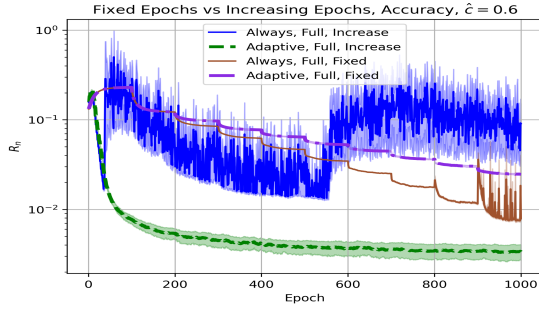


(g) PStorm, Adaptive epochs, Accuracy, $\hat{c} = 0.9$.

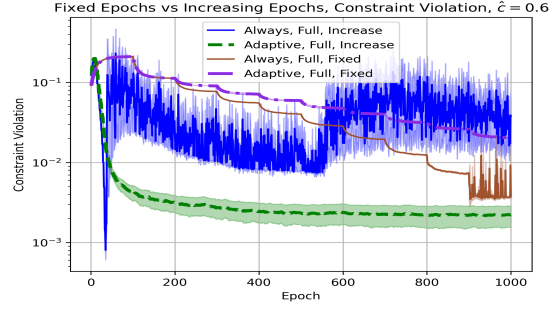


(h) PStorm, Adaptive epochs, Constr. violation, $\hat{c} = 0.9$.

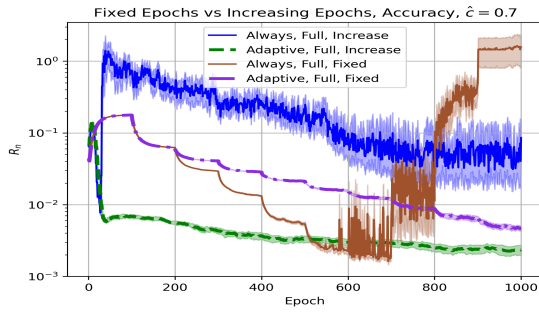
Figure 4: Comparison of different outer loop modes for PStorm with adaptive inner epochs varying $\hat{c}$. Average results over 20 trials with 95% confidence intervals.



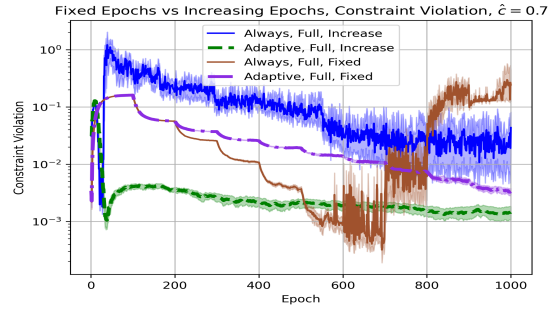
(a) PStorm, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.6$.



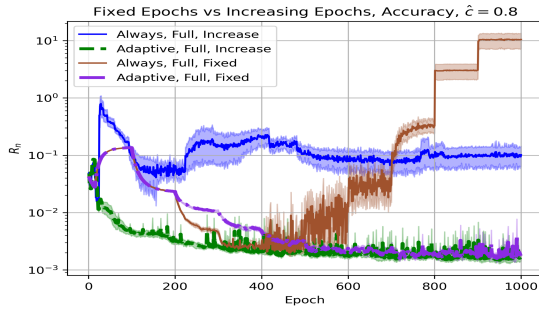
(b) PStorm, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.6$.



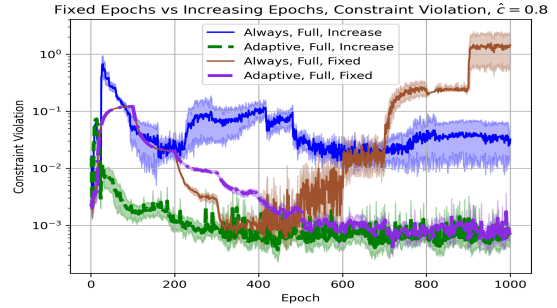
(c) PStorm, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.7$.



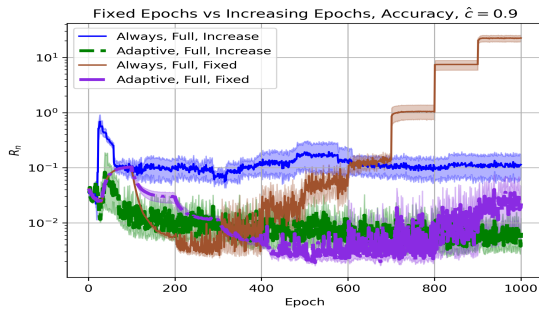
(d) PStorm, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.7$.



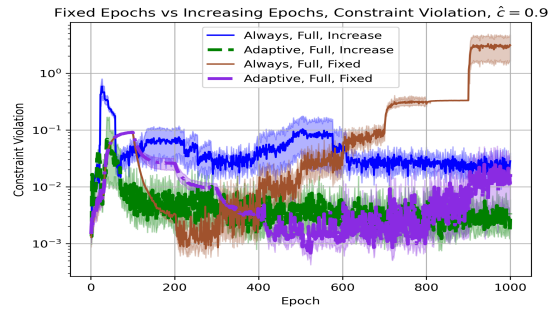
(e) PStorm, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.8$.



(f) PStorm, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.8$.



(g) PStorm, Adapt vs. Fixed, Accuracy, $\hat{c} = 0.9$.



(h) PStorm, Adapt vs. Fixed, Constr. Violation, $\hat{c} = 0.9$.

Figure 5: Comparison of adaptive versus fixed inner epochs for PStorm varying $\hat{c}$. Average results over 20 trials with 95% confidence intervals.