


# A Retrospective-Approximation Interior-Point Method

Frank E. Curtis, Zahra Khatti, and Daniel P. Robinson

Department of Industrial and Systems Engineering, Lehigh University

COR@L Technical Report 26T-022






# A Retrospective-Approximation Interior-Point Method

Frank E. Curtis*[1], Zahra Khatti†[1], and Daniel P. Robinson‡[1]

[1]Department of Industrial and Systems Engineering, Lehigh University

November 7, 2026

**Abstract**

An algorithm is proposed, analyzed, and tested for solving nonlinearly constrained continuous optimization problem instances wherein the objective function is defined by an expectation over a random variable argument and the constraints include inequalities (and perhaps equalities) involving functions that may be nonconvex. The algorithm combines a retrospective-approximation strategy for handling the objective and an interior-point approach for handling the constraints. In each iteration of an outer loop, the objective is approximated by an empirical average over a set of realizations of the random variable argument and the resulting sampled problem is solved with an interior-point method that allows inexact search-direction computations for computational flexibility. Termination conditions for the interior-point method are tailored for those needed for convergence guarantees by the outer loop. Over the outer loop, the sample set is updated adaptively in such a manner that convergence guarantees for solving the original optimization problem can be assured. The results of numerical experiments are presented, which show that the proposed algorithm can outperform competing methods that have comparable theoretical guarantees.

## 1 Introduction

The purpose of this paper is to propose, analyze, and provide the results of numerical experiments with an efficient algorithm for solving nonlinearly constrained continuous optimization problems. In particular, our proposed algorithm is designed for settings in which the objective function is defined by an expectation over a random variable argument and the inequality (and any equality) constraints are defined by functions that may be nonconvex. Multiple types of algorithms exist for solving such problems, each of which offers different types of convergence guarantees. Our focus in this paper is on a class of algorithms that aims to solve the original problem to high accuracy by sampling increasingly larger numbers of terms for approximating the objective function. At the same time, our proposed algorithm aims to exploit the fact that the original problem may be solved more efficiently by starting with a small number of sampled objective terms, employing a deterministic algorithm to solve the corresponding approximate optimization problem to moderate accuracy, and only then increasing the sample size. This leads to an iterative scheme that aims to be efficient in terms of sampling the objective function, while also eventually ensuring that the original problem of interest is solved to high accuracy.

The algorithm we propose combines a retrospective-approximation (RA) strategy for handling the objective function and an interior-point method (IPM) for handling the constraints. RA is a paradigm for solving continuous optimization problems that has been proposed and analyzed for both unconstrained [27, 28] and constrained [2] settings. It is closely related to sample-average-approximation schemes in which a sequence

*E-mail: frank.e.curtis@lehigh.edu

†E-mail: zak223@lehigh.edu

‡E-mail: daniel.p.robinson@lehigh.edu

of sampled problems with increasing sample sizes is solved, each only approximately; see, e.g., [31, 29], in which the sample size is increased according to rules based on optimality functions, and [30], in which the sample sizes are selected with the aim of minimizing the total computational effort. RA schemes that employ an IPM to solve each sampled problem have been considered for two-stage stochastic linear programs in [20]; our setting, with nonlinear and possibly nonconvex constraints, is different. On the other hand, the class of IPMs is known to be one of the most effective classes of algorithmic methods for solving large-scale continuous optimization problems. This is evidenced by IPM strategies forming the core of state-of-the-art software packages such as Ipopt [36], Knitro [9], and LOQO [34].

Our approach can be contrasted with stochastic variants of sequential quadratic programming (SQP) methods and IPMs that have been proposed for this same class of problems, i.e., with an objective defined by an expectation and nonlinear constraint functions that can be evaluated deterministically. Stochastic SQP methods that employ a stochastic objective-gradient estimate in each iteration were first proposed for problems with nonlinear equality constraints in [5], and this line of work has since been extended to allow for rank-deficient constraint Jacobians through a step-decomposition strategy [4] and to allow for inexact subproblem solutions, which, as in this paper, is of particular relevance in large-scale settings [15]. For problems with nonlinear inequality and equality constraints, stochastic SQP methods include one with convergence guarantees in expectation under loose assumptions on the gradient estimates [14] and an active-set method that employs an augmented Lagrangian merit function and a stochastic line search and offers almost-sure convergence guarantees [26]. More recently, stochastic IPMs have been proposed as well: a stochastic-gradient-based IPM for bound-constrained problems [12]; a single-loop stochastic IPM for nonlinear inequality-constrained problems that maintains feasibility of the iterates with respect to the inequality constraints while the barrier parameter is driven to zero along with the stochastic-gradient iterations [11]; and a trust-region stochastic SQP method for nonlinear inequality- and equality-constrained problems that handles the inequality constraints with an interior-point approach, decreases the barrier parameter along a prescribed sequence, and offers almost-sure convergence guarantees [18]. Methods of this type take a stochastic step in every iteration, so their step sizes and merit or barrier parameters must be controlled to account for the stochastic error present in every iteration, and they do not directly exploit mature deterministic solvers for constrained problems. A complementary paradigm is to increase the number of samples as the optimization proceeds, as in the progressive-batching and adaptive-sampling strategies developed for unconstrained problems [7, 6] and extended to equality-constrained problems in the adaptive-sampling SQP method of [3], so that the stochastic error is reduced as a solution is approached. RA can be viewed as the member of this paradigm in which each sampled problem is solved by a deterministic method to a prescribed accuracy before the sample size is increased, which is precisely the feature that allows an IPM with inexact step computations to be designed.

## 1.1 Contributions

Our proposed algorithm builds on the RA framework proposed in [2] for solving constrained optimization problems by combining it with an IPM that uses the inexact search direction computations proposed in [16, 10]. As opposed to the RA-SQP framework proposed in [2], which involves distinct termination criteria from the general RA framework analyzed in that paper, our RA-IPM strategy involves termination criteria for the inner IPM that are tied exactly to those for the outer RA framework; see upcoming (3), (7), (8), and (23). The results of our numerical experiments show that, amongst other advantages, our RA-IPM approach offers improved performance in terms of CPU time compared to the aforementioned RA-SQP algorithm on a set of larger-scale test problems.

## 1.2 Notation

We use $\mathbb{R}$ to denote the set of real numbers, $\mathbb{R}^n$ to denote the set of $n$-dimensional real vectors, and $\mathbb{R}^{n\times m}$ to denote the set of $n$-by-$m$-dimensional real matrices. Given $r \in \mathbb{R}$, we use $\mathbb{R}_{>r}$ (respectively, $\mathbb{R}_{\geq r}$) to denote the set of real numbers that are greater than $r$ (respectively, greater than or equal to $r$). We use $\mathbb{N}$ to denote the set of nonnegative integers, and for any $q \in \mathbb{N}$ with $q \geq 1$, we denote $[q] := \{1, \ldots, q\}$. Given $s \in \mathbb{R}^q$, its

$i$th element is denoted as $s^{(i)}$ for each $i \in [q]$. Given $(a, b) \in \mathbb{R}^q \times \mathbb{R}^q$, we use $a \perp b$ to denote that $a_i b_i = 0$ for all $i \in [q]$.

### 1.3 Organization

Our optimization problem formulation of interest and fundamental assumptions about the problem and our proposed algorithm are provided in §2. Our proposed algorithm is presented in §3, and our analysis of it is presented in §4. The details of a specific implementation of our proposed algorithm are given in §5, and the results of numerical experiments that employ this implementation of our proposed algorithm are given in §6. Concluding remarks are provided in §7.

## 2 Problem Formulations

Our algorithm is designed to solve nonlinearly constrained continuous optimization problems of the form

$$\min_{x \in \mathbb{R}^n} \; f(x) \text{ subject to (s.t.) } c_{\mathcal{E}}(x) = 0 \text{ and } c_{\mathcal{I}}(x) \leq 0, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $c_{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^p$, and $c_{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^q$ are continuously differentiable. More precisely, it is designed for solving problems where $f$ is defined by

$$f(x) \equiv \mathbb{E}_{\xi}[F(x, \xi)],$$

where $F : \mathbb{R}^n \times \Xi \to \mathbb{R}$, $\xi$ is a random variable defined by a probability space $(\Xi, \mathcal{F}_{\xi}, \mathbb{P}_{\xi})$, and $\mathbb{E}_{\xi}$ denotes expectation taken with respect to $\mathbb{P}_{\xi}$. Under the presumption that objective function and gradient values are not tractable to compute in practice in such a context, our proposed algorithm does not aim to solve (1) directly. Rather, as discussed shortly, it solves a sequence of approximate problems over which the objective function is approximated by an empirical average of sampled terms.

We make the following assumption about (1) throughout this work. We note in passing that this assumption could be loosened such that the stated properties of the problem functions only need to hold almost surely over an open convex set that contains the iterates generated by any run of the algorithm. However, since our proposed algorithm and convergence guarantees follow using the same series of arguments in either case, we state the assumption as follows for the sake of brevity.

**Assumption 2.1.** *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below by $f_{\inf} \in \mathbb{R}$, i.e., $f(x) \geq f_{\inf}$ for all $x \in \mathbb{R}^n$. In addition, its gradient function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous and bounded, i.e., there exists $(L_{\nabla f}, \kappa_{\nabla f}) \in \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ such that, for all $(x, \bar{x}) \in \mathbb{R}^n \times \mathbb{R}^n$, one has*

$$\|\nabla f(x) - \nabla f(\bar{x})\|_2 \leq L_{\nabla f} \|x - \bar{x}\|_2 \quad \text{and} \quad \|\nabla f(x)\|_2 \leq \kappa_{\nabla f}.$$

*The constraint functions $c_{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^p$ and $c_{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^q$ are continuously differentiable and bounded, i.e., there exists $(\kappa_{c_{\mathcal{E}}}, \kappa_{c_{\mathcal{I}}}) \in \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ such that, for all $x \in \mathbb{R}^n$, one has $\|c_{\mathcal{E}}(x)\|_2 \leq \kappa_{c_{\mathcal{E}}}$ and $\|c_{\mathcal{I}}(x)\|_2 \leq \kappa_{c_{\mathcal{I}}}$. In addition, $\nabla c_{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^{n \times p}$ and $\nabla c_{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^{n \times q}$ are Lipschitz continuous and bounded, i.e., there exist $(L_{\nabla c_{\mathcal{E}}}, L_{\nabla c_{\mathcal{I}}}, \kappa_{\nabla c_{\mathcal{E}}}, \kappa_{\nabla c_{\mathcal{I}}}) \in \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ such that, for all $(x, \bar{x}) \in \mathbb{R}^n \times \mathbb{R}^n$:*

$$\begin{aligned} &\|\nabla c_{\mathcal{E}}(x) - \nabla c_{\mathcal{E}}(\bar{x})\|_2 \leq L_{\nabla c_{\mathcal{E}}} \|x - \bar{x}\|_2; \;\; \|\nabla c_{\mathcal{E}}(x)\|_2 \leq \kappa_{\nabla c_{\mathcal{E}}}; \\ &\|\nabla c_{\mathcal{I}}(x) - \nabla c_{\mathcal{I}}(\bar{x})\|_2 \leq L_{\nabla c_{\mathcal{I}}} \|x - \bar{x}\|_2; \;\; \|\nabla c_{\mathcal{I}}(x)\|_2 \leq \kappa_{\nabla c_{\mathcal{I}}}. \end{aligned}$$

Since our framework uses an IPM to solve a sequence of approximations to (1), there are benefits of using a slack reformulation so that nonnegativity only needs to be maintained for the slack variables during a run of the IPM, rather than values of $c_{\mathcal{I}}$ explicitly. Thus, we introduce the following problem that is equivalent to (1):

$$\min_{(x,s) \in \mathbb{R}^n \times \mathbb{R}^q} \; f(x) \;\text{ s.t. }\; c_{\mathcal{E}}(x) = 0, \; c_{\mathcal{I}}(x) + s = 0, \text{ and } s \geq 0. \tag{2}$$

Under Assumption 2.1 and a constraint qualification, such as the Mangasarian-Fromovitz constraint qualification (MFCQ) [25], any local minimizer of (2) satisfies its corresponding Karush-Kuhn-Tucker (KKT) conditions, which can be stated as

$$\nabla f(x) + \nabla c_{\mathcal{E}}(x) y_{\mathcal{E}} + \nabla c_{\mathcal{I}}(x) y_{\mathcal{I}} = 0, \tag{3a}$$

$$y_{\mathcal{I}} - z = 0, \quad c_{\mathcal{E}}(x) = 0, \quad c_{\mathcal{I}}(x) + s = 0, \tag{3b}$$

$$\text{and} \quad 0 \leq s \perp z \geq 0. \tag{3c}$$

However, as is generally the case for local-search algorithms for solving nonlinearly constrained continuous optimization problems, it is also possible for the algorithm to be attracted to an infeasible point that minimizes a measure of constraint violation. We say that a point $x$ is an infeasible stationary point with respect to (1) if and only if it is infeasible for (1), yet is stationary for

$$\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2}\|c_{\mathcal{E}}(x)\|_2^2 + \tfrac{1}{2}\|\max\{c_{\mathcal{I}}(x), 0\}\|_2^2.$$

Assuming that a given point $(x, s)$ has $s \geq 0$ and $c_{\mathcal{I}}(x) + s \geq 0$, the first-order optimality conditions with respect to this feasibility problem reduce to

$$\nabla c_{\mathcal{E}}(x) c_{\mathcal{E}}(x) + \nabla c_{\mathcal{I}}(x)(c_{\mathcal{I}}(x) + s) = 0 \ \text{ and } \ S(c_{\mathcal{I}}(x) + s) = 0, \tag{4}$$

where $S := \operatorname{diag}(s) \in \mathbb{R}^{q \times q}$ has $(i,i)$-component equal to $s^{(i)}$ for all $i \in [q]$.

Our algorithm, proposed in the next section, employs a nested loop structure where the outermost loop, indexed by $k \in \mathbb{N}$, approximates the objective of (1) through sampling of the random variable input $\xi$. Specifically, each iteration $k \in \mathbb{N}$ is equipped with a prescribed sample size $N_k$ and employs a sample set

$$\mathcal{S}_k := \{\xi_{k,1}, \ldots, \xi_{k,N_k}\}, \tag{5}$$

through which it approximates $f$ with $f_{\mathcal{S}_k} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$f_{\mathcal{S}_k}(x) = \frac{1}{N_k} \sum_{i \in [N_k]} F(x, \xi_{k,i}).$$

This leads to the slack-based formulation of the sampled problem given by

$$\min_{(x,s) \in \mathbb{R}^n \times \mathbb{R}^q} \ f_{\mathcal{S}_k}(x) \ \text{ s.t. } \ c_{\mathcal{E}}(x) = 0, \ c_{\mathcal{I}}(x) + s = 0, \ \text{and } s \geq 0. \tag{6}$$

Consistent with Assumption 2.1 for the full-sample objective function $f$, we make the following assumption throughout this work, where for simplicity we assume that the bounds and Lipschitz constant for the sampled objective function are consistent with those in Assumption 2.1 for the full-sample objective function.

**Assumption 2.2.** *Let $(f_{\inf}, L_{\nabla f}, \kappa_{\nabla f}) \in \mathbb{R} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ be defined in Assumption 2.1. For all $k \in \mathbb{N}$, the sampled objective function $f_{\mathcal{S}_k} : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below by $f_{\inf}$. In addition, for all $k \in \mathbb{N}$, the gradient function $\nabla f_{\mathcal{S}_k} : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous with respect to the $l_2$-norm with constant $L_{\nabla f}$ and bounded in $l_2$-norm by $\kappa_{\nabla f}$.*

Under Assumption 2.2 and a constraint qualification such as the MFCQ, for any $k \in \mathbb{N}$ any local minimizer of (6) satisfies the corresponding KKT conditions

$$\nabla f_{\mathcal{S}_k}(x) + \nabla c_{\mathcal{E}}(x) y_{\mathcal{E}} + \nabla c_{\mathcal{I}}(x) y_{\mathcal{I}} = 0, \tag{7a}$$

$$y_{\mathcal{I}} - z = 0, \tag{7b}$$

$$c_{\mathcal{E}}(x) = 0, \tag{7c}$$

$$c_{\mathcal{I}}(x) + s = 0, \tag{7d}$$

$$\text{and} \quad 0 \leq s \perp z \geq 0. \tag{7e}$$

Note that (7a) differs from (3a), but (7b)–(7e) match (3b)–(3c) exactly.

# 3 Algorithm

Our proposed algorithm aims to solve (1) by solving a sequence of problems involving sampled objective functions. For each sampled objective function, an IPM is employed to solve approximately the corresponding instance of (6). In this section, we describe the procedures involved in the outer loop as well as the IPM employed to solve each sampled problem.

## 3.1 Retrospective Approximation Method

As mentioned, our algorithm employs a prescribed sequence of sample sizes $\{N_k\}$. The requirements that we impose on $\{N_k\}$, as well as on other algorithmic quantities introduced in this section, are stated along with our theoretical analysis in §4.

The main feature of our RA strategy is the set of termination conditions that are employed that dictate when it has been determined that a given sampled problem has been solved to sufficient accuracy and a new sample should be taken. For each $k \in \mathbb{N}$, our algorithm employs an IPM to generate, from an initial primal-dual iterate $(x_{k,0}, s_{k,0}, y_{\mathcal{E},k,0}, y_{\mathcal{I},k,0}, z_{k,0})$ and with an initial barrier parameter $\mu_{k,0}$, an approximate primal-dual solution $(x_{k,M_k}, s_{k,M_k}, y_{\mathcal{E},k,M_k}, y_{\mathcal{I},k,M_k}, z_{k,M_k})$ of (6) along with a final barrier parameter $\mu_{k,M_k}$. Here, $M_k \in \mathbb{N}$ represents the number of iterations that the IPM performs in iteration $k$; see §3.2. Importantly, as seen in the next section and in our theoretical analysis, our initialization procedure and termination conditions for the IPM ensure that these points satisfy $s_{k,0} > 0$, $s_{k,M_k} > 0$, $c_{\mathcal{I}}(x_{k,0}) + s_{k,0} \geq 0$, $c_{\mathcal{I}}(x_{k,M_k}) + s_{k,M_k} \geq 0$, $z_{k,0} > 0$, and $z_{k,M_k} > 0$. Consequently, determining whether an infeasible stationary point has been reached only requires checking (4), and all that is required to measure the error in satisfying the KKT conditions (7) is to measure the error in satisfying (7a)–(7d) and the error in satisfying the orthogonality condition in (7e). Following a similar, but distinct strategy from that in the RA-SQP [2], our algorithm employs the conditions

$$\begin{aligned} &\|\nabla f_{\mathcal{S}_k}(x_{k,M_k}) + \nabla c_{\mathcal{E}}(x_{k,M_k}) y_{\mathcal{E},k,M_k} + \nabla c_{\mathcal{I}}(x_{k,M_k}) y_{\mathcal{I},k,M_k}\|_\infty \\ &\leq \gamma_{k,1} \|\nabla f_{\mathcal{S}_k}(x_{k,0}) + \nabla c_{\mathcal{E}}(x_{k,0}) y_{\mathcal{E},k,0} + \nabla c_{\mathcal{I}}(x_{k,0}) y_{\mathcal{I},k,0}\|_\infty + \epsilon_{k,1}, \end{aligned} \tag{8a}$$

and

$$\|y_{\mathcal{I},k,M_k} - z_{k,M_k}\|_\infty \leq \gamma_{k,2} \|y_{\mathcal{I},k,0} - z_{k,0}\|_\infty + \epsilon_{k,2} \tag{8b}$$

$$\|c_{\mathcal{E}}(x_{k,M_k})\|_\infty \leq \gamma_{k,3} \|c_{\mathcal{E}}(x_{k,0})\|_\infty + \epsilon_{k,3}, \tag{8c}$$

$$\|c_{\mathcal{I}}(x_{k,M_k}) + s_{k,M_k}\|_\infty \leq \gamma_{k,4} \|c_{\mathcal{I}}(x_{k,0}) + s_{k,0}\|_\infty + \epsilon_{k,4}, \tag{8d}$$

$$\|S_{k,M_k} z_{k,M_k}\|_\infty \leq \gamma_{k,5} \|S_{k,0} z_{k,0}\|_\infty + \epsilon_{k,5}, \tag{8e}$$

where $\gamma_{k,a} \in [0, 1)$ and $\epsilon_{k,a} \in \mathbb{R}_{>0}$ for each $k \in \mathbb{N}$ and $a \in [5]$ are tolerances that ensure convergence in the outer loop over $k \in \mathbb{N}$; see §4 for further details.

We now state Algorithm 1, which presents the outer loop of our overall algorithmic approach. The main step involves calling Algorithm 2, presented in §3.2.

Our IPM in the next subsection is guaranteed, under relatively loose assumptions, to ensure that either an iterate satisfying the termination conditions (8) is obtained or that, in the limit, feasibility with respect to (6) is not attained while the error in satisfying the stationarity conditions (4) vanishes. This latter possibility is handled by inner termination conditions that we present in §3.2.

## 3.2 Interior-Point Method (IPM)

The IPM that we propose to employ for solving (approximately) each instance of (6) is essentially that proposed in [16]. (We also direct the reader to [10], which presents additional implementation features that can improve the practical performance of the algorithm. However, in our discussion here we discuss the method from [16], since it is more straightforward to present.) For completeness, we present the complete details of the method, although the main distinguishing feature of the method that we propose here and

**Algorithm 1** Retrospective-Approximation Method

**Require:** residual tolerances $\{\gamma_{k,a}\}_{k\in\mathbb{N}} \subset [0,1]$ and $\{\epsilon_{k,a}\}_{k\in\mathbb{N}} \subset \mathbb{R}_{>0}$ for each $a \in [5]$; increase tolerances $\{\nu_{s,k}\}_{k\in\mathbb{N}} \subset \mathbb{R}_{\geq 0}$ and $\{\nu_{z,k}\}_{k\in\mathbb{N}} \subset \mathbb{R}_{\geq 0}$; initial point $(x_0, s_0, y_{\mathcal{E},0}, y_{\mathcal{I},0}, z_0) \in \mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R}^p \times \mathbb{R}^q \times \mathbb{R}^q$ satisfying $s_0 > 0$, $c_{\mathcal{I}}(x_0) + s_0 \geq 0$, and $z_0 > 0$; and barrier parameter $\mu_0 \in \mathbb{R}_{>0}$

**for** $k = 0, 1, 2, \ldots$ **do**
  **if** $(x_k, s_k)$ satisfies (4) and either $c_{\mathcal{E}}(x_k) \neq 0$ or $c_{\mathcal{I}}(x_k) \not\leq 0$ **then**
    **return** infeasible stationary point $(x_k, s_k)$
  **end if**
  Sample the elements of $\mathcal{S}_k$ (see (5)) independently from the distribution of $\xi$
  Set $(x_{k,0}, s_{k,0}, y_{\mathcal{E},k,0}, y_{\mathcal{I},k,0}, z_{k,0}, \mu_{k,0}) \leftarrow (x_k, s_k, y_{\mathcal{E},k}, y_{\mathcal{I},k}, z_k, \mu_k)$
  Call Algorithm 2 to obtain $(x_{k,M_k}, s_{k,M_k}, y_{\mathcal{E},k,M_k}, y_{\mathcal{I},k,M_k}, z_{k,M_k}, \mu_{k,M_k})$ satisfying (8)
  Choose $\iota_{s,k} \in \mathbb{R}^q_{\geq 0}$ with $\|\iota_{s,k}\|_\infty \leq \nu_{s,k}$, $\iota_{z,k} \in \mathbb{R}^q_{\geq 0}$ with $\|\iota_{z,k}\|_\infty \leq \nu_{z,k}$, and $\iota_{\mu,k} \in [0,\infty)$
  Set

$$(x_{k+1}, s_{k+1}, y_{\mathcal{E},k+1}, y_{\mathcal{I},k+1}, z_{k+1}, \mu_{k+1}) \leftarrow (x_{k,M_k}, s_{k,M_k} + \iota_{s,k}, y_{\mathcal{E},k,M_k}, y_{\mathcal{I},k,M_k}, z_{k,M_k} + \iota_{z,k}, \mu_{k,M_k} + \iota_{\mu,k})$$

**end for**

that in [16] is the set of termination conditions that we employ. In particular, we introduce termination conditions that are tailored for the outer loop stated in Algorithm 1.

As is typical for an IPM, our algorithm for solving (6) employs a nested loop where an outer loop updates a barrier parameter and an inner loop aims at solving a corresponding barrier subproblem. Letting $\mu_{k,0} \in \mathbb{R}_{>0}$ denote an initial value of the barrier parameter corresponding to outer loop iteration $k$ (i.e., for solving the $k$th sampled problem (6)), a loop for updating the barrier parameter, indexed by $j \in \mathbb{N}$, has in iteration $j$ a value $\mu_{k,j} \in \mathbb{R}_{>0}$ of the barrier parameter. An inner loop, indexed by $l \in \mathbb{N}$, aims to solve the barrier subproblem

$$\begin{aligned} \min_{(x,s)\in\mathbb{R}^n\times\mathbb{R}^q} \; & f_{\mathcal{S}_k}(x) - \mu_{k,j} \sum_{i\in[q]} \ln(s^{(i)}) \\ \text{s.t. } & c_{\mathcal{E}}(x) = 0,\; c_{\mathcal{I}}(x) + s = 0,\; \text{and } s > 0. \end{aligned} \tag{9}$$

(Here, the constraint $s > 0$ is included to emphasize that the slack variables need to be positive so that the logarithms evaluate to real numbers, and that the algorithm needs to maintain positivity of the slack variables.) In iteration $l$, the algorithm has a primal-dual iterate given by $(x_{k,j,l}, s_{k,j,l}, y_{\mathcal{E},k,j,l}, y_{\mathcal{I},k,j,l}, z_{k,j,l})$. Let us now describe the search direction computation and iterate update procedures employed in the inner loop over the index $l$. If these procedures lead to an iterate that is an infeasible stationary point, then the IPM terminates. Otherwise, once (9) has been solved to sufficient accuracy, the method either invokes a decrease in the barrier parameter or termination of outer iteration $k$ in Algorithm 1.

The search direction computation in iteration $(k, j, l)$ employs scaled derivatives, as described in [16]. Specifically, let us define the scaled first-order derivative values

$$g_{k,j,l} := \begin{bmatrix} \nabla f_{\mathcal{S}_k}(x_{k,j,l}) \\ -\mu_{k,j} e \end{bmatrix} \in \mathbb{R}^{n+q} \quad \text{and} \quad J_{k,j,l} := \begin{bmatrix} \nabla c_{\mathcal{E}}(x_{k,j,l})^T & 0 \\ \nabla c_{\mathcal{I}}(x_{k,j,l})^T & S_{k,j,l} \end{bmatrix} \in \mathbb{R}^{(p+q)\times(n+q)}$$

as well as the symmetric matrix

$$W_{k,j,l} := \begin{bmatrix} H_{k,j,l} & 0 \\ 0 & S_{k,j,l} Z_{k,j,l} \end{bmatrix} \in \mathbb{R}^{(n+q)\times(n+q)} \quad \text{where} \quad H_{k,j,l} \in \mathbb{R}^{n\times n}$$

and $Z_{k,j,l} = \operatorname{diag}(z_{k,j,l})$. Here, to obtain fast local convergence near a local minimizer of (6), the symmetric matrix $H_{k,j,l}$ could be set as the Hessian of the Lagrangian of (6) with respect to the variable $x$ at the current primal-dual iterate, or an approximation of this Hessian. Moreover, in place of $S_{k,j,l} Z_{k,j,l}$, one might use $\mu_{k,j} I$; see [16]. For our purposes, since we do not focus on fast local convergence guarantees, we merely presume that a Hessian modification strategy is employed, as in [16].

Iteration $(k, j, l)$ commences by computing a normal step toward satisfaction of a linearization of the constraints at the current iterate. Letting

$$c_{k,j,l} := \begin{bmatrix} c_{\mathcal{E}}(x_{k,j,l}) \\ c_{\mathcal{I}}(x_{k,j,l}) + s_{k,j,l} \end{bmatrix},$$

the normal step subproblem is defined as

$$\min_{v \in \mathbb{R}^{n+q}} \ \tfrac{1}{2}\|c_{k,j,l} + J_{k,j,l} v\|_2^2 \ \text{ s.t. } \ \|v\|_2 \leq \omega \|J_{k,j,l}^T c_{k,j,l}\|_2, \tag{10}$$

where $\omega \in \mathbb{R}_{>0}$ is a user-defined constant. The vector $v_{k,j,l} \in \mathbb{R}^{n+q}$ computed by the algorithm does not need to solve (10) exactly. Rather, it only needs to satisfy the following loose condition, which is common in continuous optimization methods.

**Normal Step Condition** *The vector $v_{k,j,l}$ is feasible for* (10) *and, for some user-defined $\epsilon_v \in (0, 1]$, satisfies the Cauchy decrease condition*

$$\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} v_{k,j,l}\|_2 \geq \epsilon_v (\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} v_{k,j,l}^C\|_2),$$

*where $v_{k,j,l}^C$ is the Cauchy point with respect to problem* (10)*; see [16].*

The algorithm next computes a tangential step $u_{k,j,l}$, which is done through computation of the combined step $d_{k,j,l} = v_{k,j,l} + u_{k,j,l}$. In the extreme case, with

$$y_{k,j,l} := \begin{bmatrix} y_{\mathcal{E},k,j,l} \\ y_{\mathcal{I},k,j,l} \end{bmatrix},$$

the algorithm computes this step by solving the linear system of equations

$$\begin{bmatrix} W_{k,j,l} & J_{k,j,l}^T \\ J_{k,j,l} & 0 \end{bmatrix} \begin{bmatrix} d_{k,j,l} \\ \delta_{y,k,j,l} \end{bmatrix} = - \begin{bmatrix} g_{k,j,l} + J_{k,j,l}^T y_{k,j,l} \\ -J_{k,j,l} v_{k,j,l} \end{bmatrix}. \tag{11}$$

However, to allow flexibility, the algorithm merely requires $(d_{k,j,l}, \delta_{y,k,j,l})$ to be computed such that, with residuals $\rho_{k,j,l} \in \mathbb{R}^{n+q}$ and $r_{k,j,l} \in \mathbb{R}^{p+q}$ defined by

$$\begin{bmatrix} \rho_{k,j,l} \\ r_{k,j,l} \end{bmatrix} := \begin{bmatrix} W_{k,j,l} & J_{k,j,l}^T \\ J_{k,j,l} & 0 \end{bmatrix} \begin{bmatrix} d_{k,j,l} \\ \delta_{y,k,j,l} \end{bmatrix} + \begin{bmatrix} g_{k,j,l} + J_{k,j,l}^T y_{k,j,l} \\ -J_{k,j,l} v_{k,j,l} \end{bmatrix} \tag{12}$$

one of three termination tests is satisfied. These termination tests are stated in terms of three intermediate conditions. The first relates to the first residual component.

**Residual Condition** *The residual $\rho_{k,j,l}$ must satisfy, for some $\kappa \in (0, 1)$,*

$$\|\rho_{k,j,l}\|_2 \leq \kappa \min \left\{ \left\| \begin{bmatrix} g_{k,j,l} + J_{k,j,l}^T y_{k,j,l} \\ -J_{k,j,l} v_{k,j,l} \end{bmatrix} \right\|_2, \left\| \begin{bmatrix} g_{k,j,l-1} + J_{k,j,l-1}^T y_{k,j,l} \\ -J_{k,j,l-1} v_{k,j,l-1} \end{bmatrix} \right\|_2 \right\}.$$

In this Residual Condition, the first term in the minimum on the right-hand side is typical for inexact Newton-type methods. The second term is included to ensure convergence, as it ensures that in certain conditions a contraction occurs; see [16].

The second intermediate condition requires either that the tangential step is small in norm relative to the normal step or that the curvature along the tangential step is sufficiently large and the objective of a tangential step subproblem (see [16, eq. (2.12)]) is sufficiently small relative to the norm of the normal step. This latter condition, required in the convergence analysis, ensures that any increase in the subproblem objective function is at most proportional to the norm of the normal step.

**Tangential Step Condition** *The tangential step $u_{k,j,l} = d_{k,j,l} - v_{k,j,l}$ must satisfy, for some user-defined constant $\psi \in \mathbb{R}_{>0}$, either the inequality $\|u_{k,j,l}\|_2 \leq \psi \|v_{k,j,l}\|_2$ or, for some user-defined constants $\theta \in \mathbb{R}_{>0}$ and $\zeta \in \mathbb{R}_{>0}$, the inequalities*

$$\begin{aligned} & \tfrac{1}{2} u_{k,j,l}^T W_{k,j,l} u_{k,j,l} \geq \theta \|u_{k,j,l}\|_2^2 \\ \text{and} \quad & (g_{k,j,l} + W_{k,j,l} v_{k,j,l})^T u_{k,j,l} + \tfrac{1}{2} u_{k,j,l}^T W_{k,j,l} u_{k,j,l} \leq \zeta \|v_{k,j,l}\|_2. \end{aligned}$$

In order to ensure that the Tangential Step Condition may ultimately be satisfied, the algorithm employs the following Hessian modification strategy; see [16].

**Hessian Modification Strategy** *Let $W_{k,j,l}$ be the current value of this matrix employed in (11)–(12) and let $(d_{k,j,l}, \delta_{y,k,j,l})$ be a given trial step. If $\|u_{k,j,l}\|_2 \leq \psi \|v_{k,j,l}\|_2$ or $\frac{1}{2} u_{k,j,l}^T W_{k,j,l} u_{k,j,l} \geq \theta \|u_{k,j,l}\|_2^2$, where $(\psi, \theta)$ are defined within the Tangential Step Condition, then maintain this value for the matrix; otherwise, modify $W_{k,j,l}$ to increase its smallest eigenvalue by at least a positive constant amount.*

The third intermediate condition relates to a reduction in a local model of a merit function. For these purposes, with respect to any barrier parameter $\mu_{k,j}$ and a positive merit parameter $\pi$, the value of which is updated adaptively by the algorithm, we introduce the merit function $\phi_{\mathcal{S}_k}(\cdot, \cdot; \mu_{k,j}, \pi) : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}$ defined by

$$\phi_{\mathcal{S}_k}(x, s; \mu_{k,j}, \pi) = f_{\mathcal{S}_k}(x) - \mu_{k,j} \sum_{i \in [q]} \ln(s^{(i)}) + \pi \left\| \begin{bmatrix} c_{\mathcal{E}}(x) \\ c_{\mathcal{I}}(x) + s \end{bmatrix} \right\|_2.$$

Given $(\mu_{k,j}, \pi)$, a local model of this merit function at a given iterate $(x_{k,j,l}, s_{k,j,l})$ is given by $m_{k,j,l}(\cdot; \mu_{k,j}, \pi) : \mathbb{R}^{n+q} \to \mathbb{R}$ as defined by

$$m_{k,j,l}(d; \mu_{k,j}, \pi) = f_{\mathcal{S}_k}(x_{k,j,l}) - \mu_{k,j} \sum_{i \in [q]} \ln(s_{k,j,l}^{(i)}) + g_{k,j,l}^T d + \pi \|c_{k,j,l} + J_{k,j,l} d\|_2,$$

meaning that the reduction in this local model for a given $d_{k,j,l}$ can be measured by

$$\begin{aligned} \Delta m_{k,j,l}(d_{k,j,l}; \mu_{k,j}, \pi) &:= m_{k,j,l}(0; \mu_{k,j}, \pi) - m_{k,j,l}(d_{k,j,l}; \mu_{k,j}, \pi) \\ &= -g_{k,j,l}^T d_{k,j,l} + \pi(\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} d_{k,j,l}\|_2). \end{aligned}$$

The third intermediate condition can now be stated as follows, which refers to the merit parameter values $\pi_{k,j,l-1}$ and $\pi_{k,j,l}$ that are computed within the algorithm.

**Model Reduction Condition** *For some $\pi_{k,j,l} \geq \pi_{k,j,l-1} > 0$, the full step $d_{k,j,l}$ must satisfy, for some user-defined constant $\sigma \in (0, 1)$ and $\theta$ defined within the Tangential Step Condition, the inequality*

$$\begin{aligned} \Delta m_{k,j,l}(d_{k,j,l}; \mu_{k,j}, \pi_{k,j,l}) \geq \; & \max\{\tfrac{1}{2} u_{k,j,l}^T W_{k,j,l} u_{k,j,l}, \theta \|u_{k,j,l}\|_2^2\} \\ & + \sigma \pi_{k,j,l}(\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} v_{k,j,l}\|_2). \end{aligned}$$

We are now prepared to state the three termination tests that the algorithm employs for computing a full step $d_{k,j,l} = v_{k,j,l} + u_{k,j,l}$ composed of the normal step $v_{k,j,l}$ and a tangential step $u_{k,j,l}$ as well as a dual step $\delta_{y,k,j,l}$. Each test states a set of conditions that determine a step to be acceptable. The IPM thus employs an iterative method for solving (11), terminating as soon as one of the termination tests states that an acceptable primal-dual step has been obtained.

The first termination test essentially ensures that a step has been computed that corresponds to a sufficient reduction in the local model of the merit function for the most recent value of the merit parameter.

**Termination Test 1 (TT1)** *A step $(d_{k,j,l}, \delta_{y,k,j,l})$ is acceptable if the Residual Condition holds, the Tangential Step Condition holds, and the Model Reduction Condition holds with $\pi_{k,j,l} = \pi_{k,j,l-1}$.*

The second termination test is designed for situations in which the primal iterate is close to stationarity, and an update in the dual space is sufficiently productive.

**Termination Test 2 (TT2)** *If, for some user-defined constant* $\epsilon_2 \in \mathbb{R}_{>0}$*, one finds*

$$\|J_{k,j,l}^T c_{k,j,l}\|_2 \leq \epsilon_2 \|g_{k,j,l} + J_{k,j,l}^T y_{k,j,l}\|_2,$$

*then a step* $(d_{k,j,l}, \delta_{y,k,j,l}) \leftarrow (0, \delta_{y,k,j,l})$ *is acceptable if, with the normal step reset to zero, i.e.,* $v_{k,j,l} \leftarrow 0$*, the Residual Condition holds with* $d_{k,j,l} \leftarrow 0$*.*

The third termination test is designed for situations in which further progress in the primal space requires an increase in the merit parameter.

**Termination Test 3 (TT3)** *A step* $(d_{k,j,l}, \delta_{y,k,j,l})$ *is acceptable if the Residual Condition holds, the Tangential Step Condition holds, and, for some* $\epsilon_3 \in (0,1)$*,*

$$\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} d_{k,j,l}\|_2 \geq \epsilon_3 (\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} v_{k,j,l}\|_2) > 0.$$

In case an acceptable step is computed satisfying either Termination Test 1 or 2, then the algorithm sets the new merit parameter value as $\pi_{k,j,l} \leftarrow \pi_{k,j,l-1}$. Otherwise, the new merit parameter value should be set to ensure that the computed search direction is one of sufficient descent for the merit function. This is accomplished by setting, for some user-defined constant $\tau \in (0,1)$, the trial value

$$\hat{\pi}_{k,j,l} \leftarrow \frac{g_{k,j,l}^T d_{k,j,l} + \max\{\frac{1}{2} u_{k,j,l}^T W_{k,j,l} u_{k,j,l}, \theta \|u_{k,j,l}\|_2^2\}}{(1-\tau)(\|c_{k,j,l}\|_2 - \|c_{k,j,l} + J_{k,j,l} d_{k,j,l}\|_2)} \tag{13}$$

then setting the new merit parameter $\pi_{k,j,l}$ as equal to the maximum of $\pi_{k,j,l-1}$ and this trial merit parameter value plus a user-defined constant $\epsilon_\pi \in \mathbb{R}_{>0}$.

All that remains to describe iteration $(k,j,l)$ of our IPM is to describe the procedures for updating the primal-dual iterate and for testing for termination. Due to the algorithm's use of scaled derivatives, the primal step is

$$\begin{bmatrix} \tilde{d}_{x,k,j,l} \\ \tilde{d}_{s,k,j,l} \end{bmatrix} := \tilde{d}_{k,j,l} := \begin{bmatrix} d_{x,k,j,l} \\ S_{k,j,l} d_{s,k,j,l} \end{bmatrix}, \text{ where } \begin{bmatrix} d_{x,k,j,l} \\ d_{s,k,j,l} \end{bmatrix} := d_{k,j,l}$$

decomposes the computed search direction $d_{k,j,l}$ into the components corresponding to the $x$ and $s$ variables. Thus, the step sizes to be computed are also defined with respect to the search direction components $\tilde{d}_{x,k,j,l} \in \mathbb{R}^n$ and $\tilde{d}_{s,k,j,l} \in \mathbb{R}^q$.

First, as is standard for an IPM, the algorithm uses a fraction-to-the-boundary rule to maintain positivity of the slack variables. This is enforced by computing, for some user-defined constant $\eta_1 \in (0,1)$, the step-size limit

$$\alpha_{k,j,l}^{\max} = \min \left\{ 1, \min_{\left\{i : \tilde{d}_{s,k,j,l}^{(i)} < 0\right\}} \frac{-\eta_1 s_{k,j,l}^{(i)}}{\tilde{d}_{s,k,j,l}^{(i)}} \right\}, \tag{14}$$

from which it follows that $\alpha_{k,j,l} \in (0, \alpha_{k,j,l}^{\max}]$ will have

$$s_{k,j,l} + \alpha_{k,j,l} \tilde{d}_{s,k,j,l} \geq (1-\eta_1) s_{k,j,l}. \tag{15}$$

Second, the algorithm performs a backtracking line search to ensure that, for some user-defined constant $\eta_2 \in (0,1)$, the following sufficient decrease condition holds:

$$\begin{aligned} &\phi_{\mathcal{S}_k}(x_{k,j,l} + \alpha_{k,j,l} \tilde{d}_{x,k,j,l}, s_{k,j,l} + \alpha_{k,j,l} \tilde{d}_{s,k,j,l}; \mu_{k,j}, \pi_{k,j,l}) \\ &\leq \phi_{\mathcal{S}_k}(x_{k,j,l}, s_{k,j,l}; \mu_{k,j}, \pi_{k,j,l}) - \eta_2 \alpha_{k,j,l} \Delta m_{k,j,l}(d_{k,j,l}; \mu_{k,j}, \pi_{k,j,l}). \end{aligned} \tag{16}$$

Third, to ensure that the iterations maintain $c_{\mathcal{I}}(x_{k,j,l}) + s_{k,j,l} \geq 0$ for all $(k,j,l)$, the algorithm performs a slack reset, which involves the component-wise update

$$s_{k,j,l+1} \leftarrow \max\{s_{k,j,l+1}, -c_{\mathcal{I}}(x_{k,j,l+1})\}. \tag{17}$$

As for the update in the dual space, the algorithm needs to update the multipliers for the nonlinear constraints as well as the multipliers for the bounds on the slack variables. With respect to the former, the algorithm merely needs to ensure that a new stacked multiplier vector $y_{k,j,l+1}$ is computed so as to satisfy the inequality

$$\|g_{k,j,l} + J_{k,j,l}^T y_{k,j,l+1}\|_2 \leq \|g_{k,j,l} + J_{k,j,l}^T (y_{k,j,l} + \delta_{y,k,j,l})\|_2. \tag{18}$$

To be more specific for the sake of clarity, in our statement of the algorithm, we state that a step size for the multiplier vector is computed as

$$\beta_{y,k,j,l} \leftarrow \arg\min_{\beta \in [0,1]} \|g_{k,j,l} + J_{k,j,l}^T (y_{k,j,l} + \beta \delta_{y,k,j,l})\|_2. \tag{19}$$

On the other hand, for the slack multipliers, the algorithm first computes

$$\delta_{z,k,j,l} = \mu_{k,j} S_{k,j,l}^{-1} e - z_{k,j,l} - S_{k,j,l}^{-1} Z_{k,j,l} \tilde{d}_{s,k,j,l}, \tag{20}$$

then, to compute the step size $\beta_{z,k,j,l}$ satisfying the fraction-to-the-boundary rule

$$z_{k,j,l} + \beta_{z,k,j,l} \delta_{z,k,j,l} \geq (1 - \eta_1) z_{k,j,l},$$

computes the limiting value

$$\beta_{z,k,j,l} = \min \left\{ 1, \min_{\left\{ i : \delta_{z,k,j,l}^{(i)} < 0 \right\}} \frac{-\eta_1 z_{k,j,l}^{(i)}}{\delta_{z,k,j,l}^{(i)}} \right\}. \tag{21}$$

Finally, we state the termination conditions for the loop over index $l$ so that either the algorithm triggers a decrease in the barrier parameter or terminates completely so that the subsequent iteration of Algorithm 1 is reached. Since the method ensures that $s_{k,j,l} > 0$ and $z_{k,j,l} > 0$ for all $(k, j, l)$, there is no need for the conditions to check positivity of the slack variables or their corresponding multipliers. Therefore, one possibility for termination is that the algorithm verifies that (8) holds with

$$\left\{ \begin{array}{c} (x_{k,j}, s_{k,j}, y_{\mathcal{E},k,j}, y_{\mathcal{I},k,j}, z_{k,j}) \\ \text{or} \\ (x_{k,j,l}, s_{k,j,l}, y_{\mathcal{E},k,j,l}, y_{\mathcal{I},k,j,l}, z_{k,j,l}) \end{array} \right\}$$

$$\text{in place of } (x_{k,M_k}, s_{k,M_k}, y_{\mathcal{E},k,M_k}, y_{\mathcal{I},k,M_k}, z_{k,M_k}). \tag{22}$$

On the other hand, the loop over $l$ terminates if an infeasible stationary point has been reached (consistent with the termination condition in Algorithm 1) or, for some user-defined constants $(\epsilon_{\mu,1}, \epsilon_{\mu,2}, \epsilon_{\mu,3}, \epsilon_{\mu,4}, \epsilon_{\mu,5}) \in \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0}$,

$$\|\nabla f_{\mathcal{S}_k}(x_{k,j,l}) + \nabla c_{\mathcal{E}}(x_{k,j,l}) y_{\mathcal{E},k,j,l} + \nabla c_{\mathcal{I}}(x_{k,j,l}) y_{\mathcal{I},k,j,l}\|_\infty \leq \epsilon_{\mu,1} \mu_{k,j}, \tag{23a}$$

$$\|y_{\mathcal{I},k,j,l} - z_{k,j,l}\|_\infty \leq \epsilon_{\mu,2} \mu_{k,j} \tag{23b}$$

$$\|c_{\mathcal{E}}(x_{k,j,l})\|_\infty \leq \epsilon_{\mu,3} \mu_{k,j}, \tag{23c}$$

$$\|c_{\mathcal{I}}(x_{k,j,l}) + s_{k,j,l}\|_\infty \leq \epsilon_{\mu,4} \mu_{k,j}, \tag{23d}$$

$$\|S_{k,j,l} z_{k,j,l} - \mu_{k,j} e\|_\infty \leq \epsilon_{\mu,5} \mu_{k,j}, \tag{23e}$$

in which case a decrease in the barrier parameter is triggered.

**Algorithm 2** Interior-Point Method

**Inputs:** initial point $(x_{k,0}, s_{k,0}, y_{\mathcal{E},k,0}, y_{\mathcal{I},k,0}, z_{k,0})$ and barrier parameter $\mu_{k,0} \in \mathbb{R}_{>0}$
**Require:** $\omega \in \mathbb{R}_{>0}$, $\epsilon_v \in (0,1]$, $\kappa \in (0,1)$, $\psi \in \mathbb{R}_{>0}$, $\theta \in \mathbb{R}_{>0}$, $\zeta \in \mathbb{R}_{>0}$, $\sigma \in (0,1)$, $\epsilon_2 \in \mathbb{R}_{>0}$, $\epsilon_3 \in (0,1)$, $\tau \in (0,1)$, $\eta_1 \in (0,1)$, $\eta_2 \in (0,1)$, $\epsilon_\pi \in \mathbb{R}_{>0}$, $\{\epsilon_{\mu,a} \in \mathbb{R}_{>0}\}_{a\in[5]}$
**for** $j = 0, 1, 2, \ldots$ **do**
  **if** $(x_{k,j}, s_{k,j}, y_{\mathcal{E},k,j}, y_{\mathcal{I},k,j}, z_{k,j})$ satisfies (8) with (22) **then**
    Set $M_k \leftarrow j$ and **break**
  **end if**
  **if** $(x_{k,j}, s_{k,j})$ satisfies (4) and either $c_{\mathcal{E}}(x_{k,j}) \neq 0$ or $c_{\mathcal{I}}(x_{k,j}) \not\leq 0$ **then**
    Set $M_k \leftarrow j$ and **break**
  **end if**
  Set $(x_{k,j,0}, s_{k,j,0}, y_{\mathcal{E},k,j,0}, y_{\mathcal{I},k,j,0}, z_{k,j,0}) \leftarrow (x_{k,j}, s_{k,j}, y_{\mathcal{E},k,j}, y_{\mathcal{I},k,j}, z_{k,j})$
  Choose $\pi_{k,j,-1} \in \mathbb{R}_{>0}$
  **for** $l = 0, 1, 2, \ldots$ **do**
    **if** $(x_{k,j,l}, s_{k,j,l}, y_{\mathcal{E},k,j,l}, y_{\mathcal{I},k,j,l}, z_{k,j,l})$ satisfies (23) or (8) with (22) **then**
      Set $L_{k,j} = l$ and **break**
    **end if**
    **if** $(x_{k,j,l}, s_{k,j,l})$ satisfies (4) and either $c_{\mathcal{E}}(x_{k,j,l}) \neq 0$ or $c_{\mathcal{I}}(x_{k,j,l}) \not\leq 0$ **then**
      Set $L_{k,j} = l$ and **break**
    **end if**
    Compute $v_{k,j,l}$ satisfying the Normal Step Condition
    Compute $(d_{k,j,l}, \delta_{y,k,j,l})$ as an approximate solution of (11)
    **while** $(d_{k,j,l}, \delta_{y,k,j,l})$ does not satisfy any of Termination Tests 1, 2, or 3 **do**
      Apply the Hessian Modification Strategy
      Compute an improved solution $(d_{k,j,l}, \delta_{y,k,j,l})$ of (11)
    **end while**
    Set $\pi_{k,j,l} \leftarrow \pi_{k,j,l-1}$
    **if** $(d_{k,j,l}, \delta_{y,k,j,l})$ does not satisfy TT1 but satisfies TT2 **then**
      Set $d_{k,j,l} \leftarrow 0$
    **else if** $(d_{k,j,l}, \delta_{y,k,j,l})$ satisfies neither TT1 nor TT2 (so it satisfies TT3) **then**
      Compute $\widehat{\pi}_{k,j,l}$ by (13)
      **if** $\pi_{k,j,l} < \widehat{\pi}_{k,j,l}$ **then**
        Set $\pi_{k,j,l} \leftarrow \widehat{\pi}_{k,j,l} + \epsilon_\pi$
      **end if**
    **end if**
    Set $\tilde{d}_{k,j,l} \leftarrow (d_{x,k,j,l}, S_{k,j,l} d_{s,k,j,l})$
    Set $\delta_{z,k,j,l}$ by (20)
    Compute $\alpha^{\max}_{k,j,l}$ by (14)
    **for** $i = 0, 1, 2, \ldots$ until (16) holds **do**
      Set $\alpha_{k,j,l} \leftarrow 2^{-i} \alpha^{\max}_{k,j,l}$
    **end for**
    Compute $\beta_{y,k,j,l}$ by (19)
    Set $\beta_{z,k,j,l}$ by (21)
    Set $x_{k,j,l+1} \leftarrow x_{k,j,l} + \alpha_{k,j,l} \tilde{d}_{x,k,j,l}$ and $s_{k,j,l+1} \leftarrow s_{k,j,l} + \alpha_{k,j,l} \tilde{d}_{s,k,j,l}$
    Set $s_{k,j,l+1} \leftarrow \max\{s_{k,j,l+1}, -c_{\mathcal{I}}(x_{k,j,l+1})\}$
    Set $y_{\mathcal{E},k,j,l+1} \leftarrow y_{\mathcal{E},k,j,l} + \beta_{y,k,j,l} \delta_{y_{\mathcal{E}},k,j,l}$ and $y_{\mathcal{I},k,j,l+1} \leftarrow y_{\mathcal{I},k,j,l} + \beta_{y,k,j,l} \delta_{y_{\mathcal{I}},k,j,l}$
    Set $z_{k,j,l+1} \leftarrow z_{k,j,l} + \beta_{z,k,j,l} \delta_{z,k,j,l}$
  **end for**
  Set $(x_{k,j+1}, s_{k,j+1}, y_{\mathcal{E},k,j+1}, y_{\mathcal{I},k,j+1}, z_{k,j+1}) \leftarrow (x_{k,j,L_{k,j}}, s_{k,j,L_{k,j}}, y_{\mathcal{E},k,j,L_{k,j}}, y_{\mathcal{I},k,j,L_{k,j}}, z_{k,j,L_{k,j}})$
  **if** the loop over $l$ terminated because (8) holds with (22) **then**
    Set $\mu_{k,j+1} \leftarrow \mu_{k,j}$
  **else**
    Set $\mu_{k,j+1} \in (0, \mu_{k,j})$
  **end if**
**end for**

# 4 Convergence Analysis

We begin our analysis by summarizing relevant results from [16], applied to our setting of Algorithm 2, that give convergence guarantees for solving (6) for all generated $k \in \mathbb{N}$. The following theorem covers potential finite termination of the algorithm, as well as potential convergence to infeasible stationary points or to

points at which a constraint qualification fails to hold.

**Theorem 4.1.** *Consider any generated $k \in \mathbb{N}$ for which Algorithm 1 calls Algorithm 2. Suppose that Assumptions 2.1 and 2.2 hold, and that for any generated $j \in \mathbb{N}$ one has that (i) after a finite number of applications of the Hessian Modification Strategy, the matrix $W_{k,j,l}$ satisfies $W_{k,j,l} \succeq 2\theta I$ and is uniformly bounded in norm (by a constant that may depend on $k$, but not on $j$), and (ii) the iterative linear system solver can solve* (11) *to arbitrary accuracy. Moreover, suppose that if $j \to \infty$, then the barrier parameter sequence vanishes, i.e., $\{\mu_{k,j}\} \searrow 0$. Then, one of the following occurs with respect to Algorithm 2.*

*(a) For some $(j, l)$, the primal point $(x_{k,j,l}, s_{k,j,l})$ is an infeasible stationary point and the algorithm terminates finitely.*

*(b) For some $j$ and all $l$, neither the set of conditions* (23) *nor* (8) *with* (22) *hold, and the algorithm generates iterates satisfying either*

$$\lim_{l\to\infty} \|J_{k,j,l}^T c_{k,j,l}\| = 0 \quad \textit{while} \quad \|c_{k,j,l}\|_\infty \geq \min\{\epsilon_{\mu,3}, \epsilon_{\mu,4}\}\mu_{k,j} \quad \textit{for all} \quad l$$

*i.e., the iterates remain infeasible and* (4) *holds in a limiting sense, or*

$$\lim_{l\to\infty} \|J_{k,j,l}^T c_{k,j,l}\| = 0 \quad \textit{and} \quad \lim_{l\to\infty} \pi_{k,j,l} = \infty,$$

*i.e.,* (4) *holds in a limiting sense and the penalty parameter diverges. This latter event cannot occur if the sequence of constraint Jacobians $\{J_{k,j,l}\}_{l=0}^{\infty}$ is bounded and all limit points have full row rank.*

*(c) For some $j$, the termination conditions in line 4 hold, $s_{k,j} > 0$, and $z_{k,j} > 0$.*

*Proof.* By construction of Algorithm 2, the conclusions follow as in [16]; in particular, see Assumptions 3.1 and 3.3 along with Theorems 3.4 and 3.13. □

Theorem 4.1 shows that the solve of (6) for each generated $k \in \mathbb{N}$ either: terminates finitely with an infeasible stationary point; iterates indefinitely due to infeasible stationarity or lack of a constraint qualification; or terminates finitely with a point satisfying (8). With respect to Algorithm 1, the only interesting case to consider is when the latter event occurs for all generated $k \in \mathbb{N}$. Moreover, if any iterate generated by Algorithm 1 is an infeasible stationary point, then there is also nothing left to prove. Therefore, in essence, the only situation that remains interesting to consider is when Algorithm 1 generates an infinite sequence of iterates.

To proceed in our analysis of Algorithm 1, we need to introduce some additional notation due to the fact that the randomness in the selection of the sample sets causes the resulting iterate sequence to be a stochastic process. (We assume for the sake of simplicity that neither the linear system solver nor any other aspect of Algorithm 2 involves stochastic quantities. Our analysis could allow such possibilities with similar arguments, but for our purposes it is sufficient to presume that the stochastic nature of the algorithm is due solely to the sample sets.) Let the possible sequences of iterations performed by Algorithm 1 define a probability space $(\Omega, \mathcal{G}, \mathbb{P})$. Let $\mathcal{K}$ be the event that Algorithm 1 generates an infinite sequence of iterates. Let $\mathcal{G}_0$ be a $\sigma$-algebra defined by the initial conditions of Algorithm 1 and for all $k \in \mathbb{N}$ with $k > 0$ let $\mathcal{G}_k$ denote the $\sigma$-algebra defined by the initial conditions and $\{\mathcal{S}_0, \ldots, \mathcal{S}_{k-1}\}$. We can now introduce the trace $\sigma$-algebra with $\mathcal{F}_k := \mathcal{G}_k \cap \mathcal{K}$ for all $k \in \mathbb{N}$ defining the filtration $\{\mathcal{F}_k\}$, which has $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{G} \cap \mathcal{K} =: \mathcal{F}$.

Let us emphasize that under this perspective, the sample sets, iterate sequence, steps computed by the algorithm, etc. are random variables. *For example, $x_k$ for each $k \in \mathbb{N}$ is a random variable that is $\mathcal{F}_k$-measurable.* Employing this analytical framework, we make the following assumption about the gradients and their estimates. The first inequality in the assumption is common in stochastic optimization methods; see, e.g., [8]. The second inequality is similar to, e.g., [2, eq. (2.7)].

**Assumption 4.1.** *Let $(\Omega, \mathcal{G}, \mathbb{P})$, $\mathcal{K}$, $\{\mathcal{G}_k\}$, $\{\mathcal{F}_k\}$, and $\mathcal{F}$ be defined as above. There exist positive constants $\chi_{\nabla f}$ and $\chi$ such that, for all $k \in \mathbb{N}$, one has*

$$\mathbb{E}[\|\nabla f(x_k) - \nabla f_{\mathcal{S}_k}(x_k)\|_\infty^2 | \mathcal{F}_k] \leq \frac{\chi_{\nabla f}\|\nabla f(x_k)\|_\infty^2 + \chi}{|\mathcal{S}_k|}$$
$$\text{and} \quad \mathbb{E}[\|\nabla f(x_{k+1}) - \nabla f_{\mathcal{S}_k}(x_{k+1})\|_\infty^2 | \mathcal{F}_k] \leq \mathbb{E}\left[ \frac{\chi_{\nabla f}\|\nabla f(x_{k+1})\|_\infty^2 + \chi}{|\mathcal{S}_k|} \middle| \mathcal{F}_k \right].$$

For the sake of brevity, let us define for all $k \in \mathbb{N}$ the primal-dual iterate

$$b_k := (x_k, s_k, y_{\mathcal{E},k}, y_{\mathcal{I},k}, z_k).$$

Let the quantities $b$, $b_{k,0}$, and $b_{k,M_k}$ be defined similarly following the notation of Algorithm 1. Further, let us define the residual functions $\{r_a\}_{a=1}^5$ and $r_{1,\mathcal{S}}$ by

$$\begin{aligned} r_1(b) &= \nabla f(x) + \nabla c_{\mathcal{E}}(x) y_{\mathcal{E}} + \nabla c_{\mathcal{I}}(x) y_{\mathcal{I}}, \\ r_{1,\mathcal{S}}(b) &= \nabla f_{\mathcal{S}}(x) + \nabla c_{\mathcal{E}}(x) y_{\mathcal{E}} + \nabla c_{\mathcal{I}}(x) y_{\mathcal{I}}, \\ r_2(b) &= y_{\mathcal{I}} - z, \\ r_3(b) &= c_{\mathcal{E}}(x), \\ r_4(b) &= c_{\mathcal{I}}(x) + s, \\ \text{and} \quad r_5(b) &= Sz. \end{aligned} \tag{24}$$

The following lemma bounds the residual norms during each iteration.

**Lemma 4.1.** *For all $k \in \mathbb{N}$, one has that*

$$\begin{aligned} \|r_1(b_{k,M_k})\|_\infty \leq \gamma_{k,1}\|r_1(b_{k,0})\|_\infty + \epsilon_{k,1} \\ + \|\nabla f(x_{k,M_k}) - \nabla f_{\mathcal{S}_k}(x_{k,M_k})\|_\infty \\ + \gamma_{k,1}\|\nabla f(x_{k,0}) - \nabla f_{\mathcal{S}_k}(x_{k,0})\|_\infty, \end{aligned} \tag{25}$$

*whereas for all $a \in \{2,3,4,5\}$ one has that*

$$\|r_a(b_{k,M_k})\|_\infty \leq \gamma_{k,a}\|r_a(b_{k,0})\|_\infty + \epsilon_{k,a}. \tag{26}$$

*Proof.* Consider arbitrary $k \in \mathbb{N}$. The inequalities in (26) follow directly from (8b)–(8e). As for (25), one finds by the triangle inequality and (8a) that

$$\begin{aligned} \|r_1(b_{k,M_k})\|_\infty &\leq \|r_{1,\mathcal{S}_k}(b_{k,M_k})\|_\infty + \|r_1(b_{k,M_k}) - r_{1,\mathcal{S}_k}(b_{k,M_k})\|_\infty \\ &\leq \gamma_{k,1}\|r_{1,\mathcal{S}_k}(b_{k,0})\|_\infty + \epsilon_{k,1} + \|r_1(b_{k,M_k}) - r_{1,\mathcal{S}_k}(b_{k,M_k})\|_\infty \\ &\leq \gamma_{k,1}(\|r_1(b_{k,0})\|_\infty + \|r_{1,\mathcal{S}_k}(b_{k,0}) - r_1(b_{k,0})\|_\infty) + \epsilon_{k,1} \\ &\quad + \|r_1(b_{k,M_k}) - r_{1,\mathcal{S}_k}(b_{k,M_k})\|_\infty. \end{aligned}$$

Since $r_1(b) - r_{1,\mathcal{S}}(b) = \nabla f(x) - \nabla f_{\mathcal{S}}(x)$ for any $b$ and $\mathcal{S}$, the proof is complete. □

Our next lemma considers the effect of the perturbation of the slack variables and their multipliers that can occur in line 9 of Algorithm 1. (Note that the perturbation of the barrier parameter in this line of the algorithm has no effect on these quantities.)

**Lemma 4.2.** *For all $k \in \mathbb{N}$, one has that*

$$\begin{aligned} \|r_2(b_{k+1})\|_\infty &\leq \|r_2(b_{k,M_k})\|_\infty + \nu_{z,k}, \\ \|r_4(b_{k+1})\|_\infty &\leq \|r_4(b_{k,M_k})\|_\infty + \nu_{s,k}, \\ \text{and} \quad \|r_5(b_{k+1})\|_\infty &\leq \|r_5(b_{k,M_k})\|_\infty + \|s_{k,M_k}\|_\infty \nu_{z,k} + \nu_{s,k}\|z_{k,M_k}\|_\infty + \nu_{s,k}\nu_{z,k}, \end{aligned}$$

*whereas for each $a \in \{1,3\}$ one has that $r_a(b_{k+1}) = r_a(b_{k,M_k})$.*

*Proof.* The only differences between $b_{k+1}$ and $b_{k,M_k}$ occur in the slack variables and their corresponding dual variables, which have no effect on the $r_1$ and $r_3$ residual values. Hence, the equations for $a \in \{1,3\}$ hold trivially. The bounds for the residuals $r_2$ and $r_4$ follow by the triangle inequality, $s_{k+1} \leftarrow s_{k,M_k} + \iota_{s,k}$, $z_{k+1} \leftarrow z_{k,M_k} + \iota_{z,k}$, and the fact that $\iota_{s,k}$ and $\iota_{z,k}$ are chosen to satisfy $\|\iota_{s,k}\|_\infty \leq \nu_{s,k}$ and $\|\iota_{z,k}\|_\infty \leq \nu_{z,k}$, respectively. The bound for the residual $r_5$ follows from the triangle inequality, the fact that $r_5(b_{k+1}) - r_5(b_{k,M_k}) = S_{k,M_k}\iota_{z,k} + Z_{k,M_k}\iota_{s,k} + \iota_{s,k} \cdot \iota_{z,k}$, the facts that $\|\iota_{s,k}\|_\infty \leq \nu_{s,k}$ and $\|\iota_{z,k}\|_\infty \leq \nu_{z,k}$, and the fact that for two vectors of the same dimension with nonnegative entries, the $\ell_\infty$-norm of their componentwise-product is less than or equal to the product of the $\ell_\infty$-norms of the two vectors. □

Our main result employs the following technical lemma.

**Lemma 4.3.** *If $q_k \in \mathbb{R}_{\geq 0}$ for all $k \in \mathbb{N}$ has $\{q_k\} \to 0$ and $\gamma \in (0,1)$, then*

$$\left\{\sum_{k=0}^{K} q_k \gamma^{K-k}\right\} \to 0 \quad \text{as} \quad K \to \infty.$$

*Proof.* Since $\{q_k\} \to 0$, it follows that $\bar{q} := \max\{q_k : k \in \mathbb{N}\} \in \mathbb{R}_{\geq 0}$ exists and is finite. Now consider arbitrary $\underline{q} \in \mathbb{R}_{>0}$. Since $\{q_k\} \to 0$, there exists $K_{\underline{q}} \in \mathbb{N}$ such that $q_k \leq \underline{q}$ for all $k \geq K_{\underline{q}}$. For any $K \in \mathbb{N}$ with $K > K_{\underline{q}}$, one finds that

$$\begin{aligned}
\sum_{k=0}^{K} q_k \gamma^{K-k} &= \sum_{k=0}^{K_{\underline{q}}} q_k \gamma^{K-k} + \sum_{k=K_{\underline{q}}+1}^{K} q_k \gamma^{K-k} \\
&\leq \bar{q} \sum_{k=0}^{K_{\underline{q}}} \gamma^{K-k} + \underline{q} \sum_{k=K_{\underline{q}}+1}^{K} \gamma^{K-k} \\
&\leq \bar{q} \frac{\gamma^{K-K_{\underline{q}}}}{1-\gamma} + \frac{\underline{q}}{1-\gamma}. \qquad (27)
\end{aligned}$$

For any $\varepsilon \in \mathbb{R}_{>0}$, one can choose $\underline{q}$ sufficiently small relative to $1-\gamma$, then choose $K_{\underline{q}}$ such that $q_k \leq \underline{q}$ for all $k \geq K_{\underline{q}}$, and finally choose $K$ sufficiently large relative to $K_{\underline{q}}$ such that (27) is less than $\varepsilon$. Thus, the desired conclusion holds. □

We are now prepared to prove a convergence guarantee for Algorithm 1.

**Theorem 4.2.** *Suppose that Assumptions 2.1, 2.2, and 4.1 hold and $\{|\mathcal{S}_k|\} \to \infty$. Suppose that there exists $\gamma_a \in (0,1)$ such that $\gamma_{k,a} \leq \gamma_a$ for all $(k,a) \in \mathbb{N} \times [5]$, and that $\{\epsilon_{k,a}\}_{k\in\mathbb{N}} \to 0$ for all $a \in [5]$, $\{\nu_{s,k}\}_{k\in\mathbb{N}} \to 0$, and $\{\nu_{z,k}\}_{k\in\mathbb{N}} \to 0$. Finally, suppose that there exist $(\kappa_s, \kappa_z) \in \mathbb{R}_{>0} \times \mathbb{R}_{>0}$ and $k_0 \in \mathbb{N}$ such that $\|s_{k,M_k}\|_\infty \leq \kappa_s$ and $\|z_{k,M_k}\|_\infty \leq \kappa_z$ for all $k \geq k_0$. Then, the sequences of expected residuals satisfy*

$$\lim_{k\to\infty} \mathbb{E}[\|r_a(b_k)\|_\infty] = 0 \quad \text{for all} \quad a \in [5].$$

*Proof.* Throughout the proof, let us define the constant

$$\kappa_\chi := \sqrt{\chi_{\nabla f} \kappa_{\nabla f}^2 + \chi} \in \mathbb{R}_{>0}.$$

Let us begin by considering the residual $r_1$. Consider arbitrary $k \in \mathbb{N}$. By Lemma 4.2, one has $r_1(b_{k+1}) = r_1(b_{k,M_k})$, so Lemma 4.1 along with the facts that $b_{k,0} = b_k$ and $x_{k,M_k} = x_{k+1}$ yields

$$\begin{aligned}
\|r_1(b_{k+1})\|_\infty \leq \gamma_{k,1}\|r_1(b_k)\|_\infty + \epsilon_{k,1} \\
+ \|\nabla f(x_{k+1}) - \nabla f_{\mathcal{S}_k}(x_{k+1})\|_\infty + \gamma_{k,1}\|\nabla f(x_k) - \nabla f_{\mathcal{S}_k}(x_k)\|_\infty.
\end{aligned}$$

Since $b_k$ is $\mathcal{F}_k$-measurable and all terms are nonnegative, one obtains that

$$\begin{aligned}\mathbb{E}[\|r_1(b_{k+1})\|_\infty|\mathcal{F}_k] \le \gamma_{k,1}\|r_1(b_k)\|_\infty + \epsilon_{k,1} \\ + \mathbb{E}[\|\nabla f(x_{k+1}) - \nabla f_{\mathcal{S}_k}(x_{k+1})\|_\infty|\mathcal{F}_k] \\ + \gamma_{k,1}\mathbb{E}[\|\nabla f(x_k) - \nabla f_{\mathcal{S}_k}(x_k)\|_\infty|\mathcal{F}_k].\end{aligned} \tag{28}$$

Let us now bound the last two terms on the right-hand side of (28). By Jensen's inequality (applied with concave $t \mapsto \sqrt{t}$), the first inequality in Assumption 4.1, and the bound $\|\nabla f(x_k)\|_\infty \le \|\nabla f(x_k)\|_2 \le \kappa_{\nabla f}$ from Assumption 2.1, one finds

$$\begin{aligned}\mathbb{E}[\|\nabla f(x_k) - \nabla f_{\mathcal{S}_k}(x_k)\|_\infty|\mathcal{F}_k] &\le \left(\mathbb{E}[\|\nabla f(x_k) - \nabla f_{\mathcal{S}_k}(x_k)\|_\infty^2|\mathcal{F}_k]\right)^{1/2} \\ &\le \left(\frac{\chi\nabla_f\|\nabla f(x_k)\|_\infty^2 + \chi}{|\mathcal{S}_k|}\right)^{1/2} \le \frac{\kappa_\chi}{\sqrt{|\mathcal{S}_k|}}.\end{aligned}$$

Similarly, by the second inequality in Assumption 4.1 and since $\|\nabla f(x_{k+1})\|_\infty \le \kappa_{\nabla f}$ holds surely by Assumption 2.1, one finds that

$$\begin{aligned}&\mathbb{E}[\|\nabla f(x_{k+1}) - \nabla f_{\mathcal{S}_k}(x_{k+1})\|_\infty|\mathcal{F}_k] \\ \le\ &\left(\mathbb{E}[\|\nabla f(x_{k+1}) - \nabla f_{\mathcal{S}_k}(x_{k+1})\|_\infty^2|\mathcal{F}_k]\right)^{1/2} \\ \le\ &\left(\mathbb{E}\left[\frac{\chi\nabla_f\|\nabla f(x_{k+1})\|_\infty^2 + \chi}{|\mathcal{S}_k|}\middle|\mathcal{F}_k\right]\right)^{1/2} \le \frac{\kappa_\chi}{\sqrt{|\mathcal{S}_k|}}.\end{aligned}$$

Combining these bounds with (28) and taking total expectation, it follows that

$$\mathbb{E}[\|r_1(b_{k+1})\|_\infty] \le \gamma_{k,1}\mathbb{E}[\|r_1(b_k)\|_\infty] + q_k, \quad \text{where} \quad q_k := \epsilon_{k,1} + (1+\gamma_{k,1})\frac{\kappa_\chi}{\sqrt{|\mathcal{S}_k|}}. \tag{29}$$

Since $k \in \mathbb{N}$ was arbitrary, (29) holds for all $k \in \mathbb{N}$. Applying it recursively, one finds for any $K \in \mathbb{N}$, with the convention that $\prod_{i=K+1}^K \gamma_{i,1} := 1$, that

$$\begin{aligned}\mathbb{E}[\|r_1(b_{K+1})\|_\infty] &\le \left(\prod_{k=0}^K \gamma_{k,1}\right)\|r_1(b_0)\|_\infty + \sum_{k=0}^K\left(\prod_{i=k+1}^K \gamma_{i,1}\right)q_k \\ &\le \gamma_1^{K+1}\|r_1(b_0)\|_\infty + \sum_{k=0}^K q_k\gamma_1^{K-k},\end{aligned} \tag{30}$$

where the second inequality follows since $\gamma_{k,1} \le \gamma_1$ and $q_k \ge 0$ for all $k \in \mathbb{N}$. The first term on the right-hand side of (30) vanishes as $K \to \infty$ since $\gamma_1 \in (0,1)$ and $b_0$ is fixed. As for the second term, since $\{\epsilon_{k,1}\} \to 0$, $\{|\mathcal{S}_k|\} \to \infty$, and $\gamma_{k,1} \le 1$ for all $k \in \mathbb{N}$, one has $\{q_k\} \to 0$, so this term vanishes as $K \to \infty$ by Lemma 4.3. This proves the desired conclusion for $a = 1$.

Now consider $a \in \{2,3,4,5\}$. Combining Lemmas 4.1 and 4.2 with the facts that $b_{k,0} = b_k$ for all $k \in \mathbb{N}$ and that $\|s_{k,M_k}\|_\infty \le \kappa_s$ and $\|z_{k,M_k}\|_\infty \le \kappa_z$ for all $k \ge k_0$,

$$\begin{aligned}&\|r_2(b_{k+1})\|_\infty \le \gamma_{k,2}\|r_2(b_k)\|_\infty + \epsilon_{k,2} + \nu_{z,k} \text{ for all } k \in \mathbb{N}, \\ &\|r_3(b_{k+1})\|_\infty \le \gamma_{k,3}\|r_3(b_k)\|_\infty + \epsilon_{k,3} \text{ for all } k \in \mathbb{N}, \\ &\|r_4(b_{k+1})\|_\infty \le \gamma_{k,4}\|r_4(b_k)\|_\infty + \epsilon_{k,4} + \nu_{s,k} \text{ for all } k \in \mathbb{N}, \\ \text{and}\ \ &\|r_5(b_{k+1})\|_\infty \le \gamma_{k,5}\|r_5(b_k)\|_\infty + \epsilon_{k,5} + \kappa_s\nu_{z,k} + \kappa_z\nu_{s,k} + \nu_{s,k}\nu_{z,k} \text{ for all } k \ge k_0.\end{aligned}$$

Unlike (29), these inequalities hold surely, i.e., for every realization of the sample sets. Each has the form

$$\|r_a(b_{k+1})\|_\infty \le \gamma_{k,a}\|r_a(b_k)\|_\infty + q_{a,k} \quad \text{for all} \quad k \ge k_a, \tag{31}$$

where $\{q_{a,k}\}$ is a deterministic nonnegative sequence with $\{q_{a,k}\} \to 0$ under the conditions of the theorem, and where $k_a := 0$ for $a \in \{2,3,4\}$ and $k_5 := k_0 + 1$. Moreover, the residual at index $k_a$ is bounded by a deterministic constant $\kappa_{r_a} \in \mathbb{R}_{\geq 0}$: for $a \in \{2,3,4\}$ one can take $\kappa_{r_a} := \|r_a(b_0)\|_\infty$ since $b_0$ is fixed, whereas for $a = 5$ it follows from line 9 of Algorithm 1 that $\|s_{k_0+1}\|_\infty \leq \kappa_s + \nu_{s,k_0}$ and $\|z_{k_0+1}\|_\infty \leq \kappa_z + \nu_{z,k_0}$, so that one can take $\kappa_{r_5} := (\kappa_s + \nu_{s,k_0})(\kappa_z + \nu_{z,k_0}) \geq \|S_{k_0+1} z_{k_0+1}\|_\infty = \|r_5(b_{k_0+1})\|_\infty$. Hence, applying (31) recursively as in (30), one finds surely for any $K \in \mathbb{N}$ with $K \geq k_a$ that

$$\begin{aligned}\|r_a(b_{K+1})\|_\infty &\leq \gamma_a^{K+1-k_a} \kappa_{r_a} + \sum_{k=k_a}^{K} q_{a,k} \gamma_a^{K-k} \\ &= \gamma_a^{K+1-k_a} \kappa_{r_a} + \sum_{m=0}^{K-k_a} q_{a,m+k_a} \gamma_a^{(K-k_a)-m}.\end{aligned}$$

The right-hand side is deterministic, so the same bound holds for $\mathbb{E}[\|r_a(b_{K+1})\|_\infty]$. The first term vanishes as $K \to \infty$ since $\gamma_a \in (0,1)$, and the second term vanishes by Lemma 4.3 applied to the nonnegative sequence $\{q_{a,m+k_a}\}_{m \in \mathbb{N}}$, which converges to zero. This proves the desired conclusion for $a \in \{2,3,4,5\}$. □

# 5 Implementation

The purpose of this section is to describe the details of an implementation of the algorithm proposed in §3, namely, that presented as Algorithms 1 and 2. Let us refer to our particular implementation as the retrospective approximation interior-point method (RA-IPM). The details to be described are the techniques for prescaling, choosing the sample sizes $\{N_k\}$, selecting the perturbations $\{(\iota_{s,k}, \iota_{z,k}, \iota_{\mu,k})\}$, computing the normal steps $\{v_{k,j,l}\}$, computing the approximate solutions of (11), and performing the line searches and iterate updates. The values of the input parameters employed are provided in Tables 1 and 2 at the end of this section. The software environment used for our experiments is described in §6.

## 5.1 Prescaling

During initialization, a problem is prescaled by the gradient-based strategy employed in Ipopt [36]. For example, the objective is multiplied by $\max\{s_{\min}^{\text{scale}}, \min\{1, g_{\max}/\|\nabla f(x_0)\|_\infty\}\}$, where $g_{\max}$ and $s_{\min}^{\text{scale}}$ are user-defined. Also, each constraint function is multiplied by an analogous factor determined by the corresponding row of the constraint Jacobians (i.e., $\nabla c_{\mathcal{E}}$ and $\nabla c_{\mathcal{I}}$) at $x_0$. These factors are held fixed throughout the solve. (Also, the same factors are supplied to every method compared in §6.) All tolerances reported below refer to the scaled problem.

## 5.2 Sampling

Algorithm 1 requires a sequence of samples $\{\mathcal{S}_k\}$ with sample sizes $\{N_k\}$. Rather than prescribe the sequence of sample sizes in advance, our implementation generates it adaptively following the variance-based rules proposed in [2, 6]. We describe this procedure in this subsection. Throughout, we presume that the optimization problem of interest has a full set $\mathcal{S}_{\text{full}}$ of $N$ realizations of $\xi$, which is generated prior to the first iteration (in the same manner for all methods in §6.) Every sample employed by the implementation is therefore a subset of $\mathcal{S}_{\text{full}}$ that is drawn without replacement, subject to an acceptance rule explained in §6.

Let us now describe the procedure for selecting the sample sizes. For any sample $\mathcal{S}$, let $r_{\mathcal{S}}(b)$ denote the stacked vector of residuals as defined in (24) (using $r_{1,\mathcal{S}}$, not $r_1$). In iteration $k = 0$, an initial sample size $N_0$ is employed. Otherwise, for $k > 0$, at the beginning of the iteration, a subset $\widetilde{\mathcal{S}}_{k-1} \subseteq \mathcal{S}_{\text{full}}$ of size $\widetilde{N}_{k-1} = N_{k-1}$ is drawn. This draw is made independently of $\mathcal{S}_{k-1}$, since $x_{k,0}$ inherits information from the inner solve performed over $\mathcal{S}_{k-1}$, meaning that a variance estimated over $\mathcal{S}_{k-1}$ would be optimistic [2]. Over

$\widetilde{\mathcal{S}}_{k-1}$, the implementation computes the sample variance

$$\mathrm{Var}_{\widetilde{\mathcal{S}}_{k-1}}(x_{k,0}) := \frac{1}{\widetilde{N}_{k-1}} \sum_{\xi \in \widetilde{\mathcal{S}}_{k-1}} \left\| \nabla F(x_{k,0}, \xi) - \nabla f_{\widetilde{\mathcal{S}}_{k-1}}(x_{k,0}) \right\|_2^2, \tag{32}$$

and forms the sample size for which the standard deviation of the sampled gradient would be a fraction $\tilde{\theta} \in (0,1)$ of $\|r_{\widetilde{\mathcal{S}}_{k-1}}(b_{k,0})\|_2$, namely, the target

$$N_k^{\mathrm{var}} := \left\lceil \frac{\mathrm{Var}_{\widetilde{\mathcal{S}}_{k-1}}(x_{k,0})}{\tilde{\theta}^2 \|r_{\widetilde{\mathcal{S}}_{k-1}}(b_{k,0})\|_2^2} \right\rceil. \tag{33}$$

The target $N_k^{\mathrm{var}}$ is determined entirely by the quantities estimated over $\widetilde{\mathcal{S}}_{k-1}$, and as such may fall below $N_{k-1}$. A floor and a ceiling are therefore imposed, which yields

$$N_k := \min\left\{N,\ \hat{\beta} N_{k-1},\ \max\{2N_{k-1},\ N_k^{\mathrm{var}}\}\right\}, \tag{34}$$

where $\hat{\beta} \geq 2$ is a user-defined constant. Note that the presence of $2N_{k-1}$ in the max forces the sample size to at least double in each outer iteration so that the full sample is reached within a budget, call it $\bar{K}$, of iterations; see Tables 1–2. The ceiling of $\hat{\beta} N_{k-1}$, on the other hand, prevents a single large variance estimate from causing the sample to reach the full sample set $\mathcal{S}_{\mathrm{full}}$ in a single large jump in sample size.

The set $\mathcal{S}_k$ is then formed by augmenting $\widetilde{\mathcal{S}}_{k-1}$ with $N_k - \widetilde{N}_{k-1}$ further realizations drawn from $\mathcal{S}_{\mathrm{full}} \setminus \widetilde{\mathcal{S}}_{k-1}$ so that $\widetilde{\mathcal{S}}_{k-1} \subset \mathcal{S}_k$. The gradient over $\mathcal{S}_k$ is recomputed rather than assembled from (32). Once $N_k = N$ is attained, the sampled problem coincides with (1) and the outer loop has no further sample to choose. At this time, the relative tolerances $\{\gamma_{k,a}\}$ in (8) are set to zero, and what remains is a single solve of (1) to the absolute tolerances. We remark that, $\mathcal{S}_{\mathrm{full}}$ being finite, this occurs after finitely many outer iterations rather than in the limit presumed in §4.

### 5.3 Warm-starting

Algorithm 1 permits the slack variables, their corresponding dual variables, and the barrier parameter to be increased before the subsequent sampled problem is solved. Consecutive sampled problems differ only through their samples, so their solutions might lie close to one another, meaning that a warm start of Algorithm 2 could be effective. That said, a barrier parameter carried forward may nonetheless be too small for a central path that has moved, and slack variables and/or their duals close to zero by the previous barrier subproblem might leave the next one ill-conditioned at a point that is no longer nearly optimal. Whether the previous solution is worth reusing thus depends on how far the sampled objective has moved. Once $\mathcal{S}_k$ has been drawn, we compute the relative gradient drift

$$\rho_k := \frac{\|\nabla f_{\mathcal{S}_k}(x_{k,0}) - \nabla f_{\mathcal{S}_{k-1}}(x_{k,0})\|_\infty}{\max\{\|\nabla f_{\mathcal{S}_{k-1}}(x_{k,0})\|_\infty,\ \underline{\rho}\}}, \tag{35}$$

where $\underline{\rho} \in \mathbb{R}_{>0}$ safeguards the denominator. (With the value $\underline{\rho} = 1$ in Table 1, $\rho_k$ measures the drift relative to the larger of the gradient norm and one, so that the test is on the absolute drift when $\|\nabla f_{\mathcal{S}_{k-1}}(x_{k,0})\|_\infty \leq 1$.) A small value of $\rho_k$ suggests that the central path of the new sampled problem lies close (at least locally) to the one that the inner loop has just followed, whereas a large value suggests that it is not. For a user-defined threshold $\bar{\rho} \in \mathbb{R}_{>0}$, we say that the transition into outer iteration $k \geq 1$ is *warm-started* when $\rho_k \leq \bar{\rho}$ and the preceding call to Algorithm 2 terminated through (8). The perturbations $\iota_{s,k-1}$ and $\iota_{z,k-1}$ chosen at the end of outer iteration $k-1$ and the barrier parameter $\mu_{k,0}$ are both governed by this test. The primal-dual iterate $(x_{k,M_k}, s_{k,M_k}, y_{\mathcal{E},k,M_k}, y_{\mathcal{I},k,M_k}, z_{k,M_k})$ is carried forward in every case, as line 9 of Algorithm 1 prescribes. A transition that is not warm-started preserves only the barrier parameter and the barrier-stage budget. Other strategies are possible. The one described here worked well in our tests.

Within Algorithm 2, our implementation decreases the barrier parameter by the monotone rule employed in Ipopt [36], namely,

$$\mu_{k,j+1} \leftarrow \max\{\underline{\mu}, \min\{\kappa_\mu \mu_{k,j}, \mu_{k,j}^{\theta_\mu}\}\} \tag{36}$$

for user-defined $\kappa_\mu \in (0,1)$, $\theta_\mu \in (1,2)$, and $\underline{\mu} \in \mathbb{R}_{>0}$, and it performs at most $\bar{J}$ barrier stages, i.e., iterations of the loop over $j$, per sampled problem.

For the barrier parameter, for a user-defined $c_\mu \geq 1$, the implementation sets

$$\mu_{k,0} \leftarrow \begin{cases} \min\{c_\mu \mu_{k-1,M_{k-1}}, \mu_0\}, & \text{if outer iteration } k \text{ is warm-started,} \\ \mu_0, & \text{otherwise,} \end{cases} \tag{37}$$

where $\mu_{k-1,M_{k-1}}$ denotes the barrier parameter upon termination of outer iteration $k-1$. Both cases are permitted since $\iota_{\mu,k-1} \in [0,\infty)$ is required there and the barrier parameter is nonincreasing within Algorithm 2, so $\mu_{k-1,M_{k-1}} \leq \mu_0$. The factor $c_\mu$ loosens the carried value, since the central path has moved even when the two samples are close. The cap at $\mu_0$ keeps the carried value no larger than the one with which outer iteration $k = 0$ begins. A warm-started iteration thus begins at a barrier parameter no larger than $\mu_0$, so the $\bar{J}$ updates available to Algorithm 2 would carry it below the value that $\bar{J}$ updates from $\mu_0$ would reach. The number of stages available in outer iteration $k$ is therefore reduced by the number of applications of (36) that would be needed to bring $\mu_0$ to or below $\mu_{k,0}$, so that the two trajectories terminate at comparable values of the barrier parameter.

As for the slack variables, a perturbation of $s$ alone leaves $x$, $y_{\mathcal{E}}$, $y_{\mathcal{I}}$, and $z$ unchanged, so by Lemma 4.2 its only effect is on the residuals $r_4$ and $r_5$, to which it contributes $\iota_{s,k}$ and $Z_{k,M_k}\iota_{s,k}$, respectively. Requiring each contribution to consume at most a fraction $c_s \in \mathbb{R}_{>0}$ of the corresponding tolerance yields the budget

$$\tilde{\epsilon}_{s,k} := c_s \min\left\{\epsilon_{k,4}, \ \frac{\epsilon_{k,5}}{\|z_{k,M_k}\|_\infty}\right\}. \tag{38}$$

Letting $\underline{s}_k := \min_{i\in[q]} s_{k,M_k}^{(i)}$ denote the smallest slack variable returned by the inner solve, we raise to the level $\underline{s}_k + \tilde{\epsilon}_{s,k}$ every slack variable lying below it, i.e., we set

$$\iota_{s,k}^{(i)} := \begin{cases} \max\{\underline{s}_k + \tilde{\epsilon}_{s,k} - s_{k,M_k}^{(i)}, \ 0\}, & \text{if outer iteration } k \text{ is warm-started,} \\ 0, & \text{otherwise,} \end{cases} \tag{39}$$

for all $i \in [q]$; equivalently, $s_{k+1}^{(i)}$ is the larger of $s_{k,M_k}^{(i)}$ and this floor. The floor is thus set relative to the smallest slack variable rather than at an absolute threshold, so that on a warm-started iteration the component attaining $\underline{s}_k$ is raised by exactly $\tilde{\epsilon}_{s,k}$, no component is raised by more, and a component already above the floor is left unchanged. One therefore has

$$\begin{aligned} \|\iota_{s,k}\|_\infty = \tilde{\epsilon}_{s,k} &\leq c_s \epsilon_{k,4} \\ \text{and} \quad \|Z_{k,M_k}\iota_{s,k}\|_\infty &\leq \|z_{k,M_k}\|_\infty \|\iota_{s,k}\|_\infty \leq c_s \epsilon_{k,5}, \end{aligned}$$

which are the conditions required in line 9 of Algorithm 1. The condition in the first line holds with equality, whereas the second line may be strict, since the largest component of $z_{k,M_k}$ and the largest component of $\iota_{s,k}$ need not be the same. Finally, the multipliers corresponding to the bounds on the slack variables are left unchanged.

## 5.4 Normal Step Computation

The normal step is computed by the dogleg procedure stated as Algorithm 3, in which the Newton step is obtained by solving the normal equations $J_{k,j,l}J_{k,j,l}^T \varpi_{k,j,l} = -c_{k,j,l}$ using a matrix factorization. Should the factorization fail, as it might for rank-deficient $J_{k,j,l}$, a positive-definite term $\delta I$ is added to $J_{k,j,l}J_{k,j,l}^T$ and

increased by a factor $\varsigma_v$ until it succeeds. In such a case, the Newton step solves a regularized problem, and on its own need not reduce linearized infeasibility by as much as the Cauchy point. The step returned is therefore taken from the segment joining the two, over which the Cauchy point is attained at $a^\star = 0$, so that the Normal Step Condition of §3.2 holds with $\epsilon_v = 1$. The scaling $\omega$ is increased by a factor of ten, to at most $\omega_{\max}$, whenever a full step is accepted whose normal component lies on the trust-region boundary, and is reset at the start of each sampled problem [10].

**Algorithm 3** Normal Step Computation

**Given** $\omega \in \mathbb{R}_{>0}$, $\delta_v \in \mathbb{R}_{>0}$ and $\varsigma_v \in \mathbb{R}_{>1}$
**if** $J_{k,j,l}^T c_{k,j,l} = 0$ **then return** $v_{k,j,l} \leftarrow 0$
Set $\Omega_{k,j,l} \leftarrow \omega \| J_{k,j,l}^T c_{k,j,l} \|_2$
Set the Cauchy point $v_{k,j,l}^C \leftarrow -\min\left\{\omega,\ \dfrac{\|J_{k,j,l}^T c_{k,j,l}\|_2^2}{\|J_{k,j,l} J_{k,j,l}^T c_{k,j,l}\|_2^2}\right\} J_{k,j,l}^T c_{k,j,l}$
Set $\delta \leftarrow 0$
**loop**
    Attempt to factor $J_{k,j,l} J_{k,j,l}^T + \delta I$ and solve $(J_{k,j,l} J_{k,j,l}^T + \delta I)\varpi_{k,j,l} = -c_{k,j,l}$
    **if** successful **then break**; **else** set $\delta \leftarrow \delta_v$ if $\delta = 0$ and $\delta \leftarrow \varsigma_v \delta$ otherwise
**end loop**
Set $v_{k,j,l}^N \leftarrow J_{k,j,l}^T \varpi_{k,j,l}$
**if** $\|v_{k,j,l}^C\|_2 = \Omega_{k,j,l}$ **then**
    **return** $v_{k,j,l} \leftarrow v_{k,j,l}^C$
**else**
    Compute $a^\star$ as a solution of

$$\min_{a \in [0,1]} \ \|c_{k,j,l} + J_{k,j,l}(v_{k,j,l}^C + a(v_{k,j,l}^N - v_{k,j,l}^C))\|_2 \quad \text{s.t.} \quad \|v_{k,j,l}^C + a(v_{k,j,l}^N - v_{k,j,l}^C)\|_2 \le \Omega_{k,j,l}$$

    **return** $v_{k,j,l} \leftarrow v_{k,j,l}^C + a^\star(v_{k,j,l}^N - v_{k,j,l}^C)$
**end if**

## 5.5 Primal-Dual Step Computation

Algorithm 2 requires $(d_{k,j,l}, \delta_{y,k,j,l})$ to be an approximate solution of (11) satisfying one of Termination Tests 1, 2, or 3 of §3.2. Our implementation computes such a step with an iterative method, described below, that ultimately tests every iterate and accepts the first one that passes a termination test. As preconditioner we employ the constraint preconditioner obtained from (11) by replacing the leading block with $D_{k,j,l} := \operatorname{diag}(\max\{|\operatorname{diag}(W_{k,j,l})|, \delta_P\})$; see [24]. Its Schur complement $J_{k,j,l} D_{k,j,l}^{-1} J_{k,j,l}^T$ is approximated by dropping each off-diagonal entry that falls below a relative tolerance $\tau_P$ and adding the magnitude of each dropped entry to the two corresponding diagonal entries, which preserves positive definiteness, and by adding $\delta_P$ to every diagonal entry, which amounts to replacing the zero lower right block of (11) by $-\delta_P I$. On problems for which that approximation would be denser than the block system itself, the latter is factored directly instead. This preconditioner is symmetric but indefinite, so a GMRES-type method is employed. Within (11) we set $H_{k,j,l}$ to be the Hessian of the Lagrangian of (6) with respect to $x$ at the current primal-dual iterate, and we retain the primal-dual block $S_{k,j,l} Z_{k,j,l}$ rather than employ $\mu_{k,j} I$, subject to

$$s_{k,j,l}^{(i)} z_{k,j,l}^{(i)} \in [\underline{\kappa}_{sz} \mu_{k,j}, \overline{\kappa}_{sz} \mu_{k,j}] \quad \text{for all} \quad i \in [q],$$

as in the default Ipopt implementation [36].

We apply the preconditioner described above on the right, within unrestarted flexible GMRES (FGMRES) [32]. Writing (11) as $\mathcal{A} w = h$, right preconditioning minimizes the unpreconditioned residual norm $\|h - \mathcal{A} w\|_2$, which is the norm bounded in (40), and FGMRES stores the preconditioned basis vectors, so that each iterate is available to the termination tests without an additional application of the preconditioner. With left preconditioning, GMRES would instead minimize $\|\mathcal{P}^{-1}(h - \mathcal{A} w)\|_2$, where $\mathcal{P}$ denotes the preconditioner. Since FGMRES forms its iterates from the preconditioned basis vectors, which need not be well conditioned, their attainable accuracy can be limited. We therefore compare the residual $\mathcal{R}_t := \|h - \mathcal{A} w_t\|_2$

of the iterate $w_t$, which the termination tests evaluate in any case, with the estimate $\widehat{\mathcal{R}}_t$ that FGMRES maintains in iteration $t$ and which equals it in exact arithmetic. If $\mathcal{R}_t > \max\{\kappa_{\mathrm{gap}}\widehat{\mathcal{R}}_t, \epsilon_{\mathrm{gap}}\|h\|_2\}$, or if the Arnoldi process breaks down without an accepted iterate, the solve switches once to left-preconditioned GMRES [33], whose iterates are formed from an orthonormal basis, starting from the FGMRES iterate with the smallest $\mathcal{R}_t$. In either phase, with $t$ counting the iterations of that phase, the solve is abandoned if $\mathcal{R}_t$ has failed to halve over the last $\max\{l^{\mathrm{pd}}_{\mathrm{stall}},\ \lfloor (t-1)/2 \rfloor\}$ iterations. The iteration limit $l^{\mathrm{pd}}_{\mathrm{max}}$ applies to the two phases together.

To avoid an unnecessarily large number of inner iterations, Algorithm 2 aims to produce reasonably accurate solutions that satisfy the residual bound

$$\left\| \begin{bmatrix} \rho_{k,j,l} \\ r_{k,j,l} \end{bmatrix} \right\|_2 \leq \kappa_{\mathrm{des}} \left\| \begin{bmatrix} g_{k,j,l} + J_{k,j,l}^T y_{k,j,l} \\ -J_{k,j,l} v_{k,j,l} \end{bmatrix} \right\|_2 \tag{40}$$

for a given $\kappa_{\mathrm{des}} \in (0,1)$ before consideration of the termination tests; see [16]. This bound differs from the Residual Condition of §3.2 in that it constrains both residual blocks and is relative to the current right-hand side alone. However, if after a fixed number of iterations $\tilde{l}^{\mathrm{pd}}_{\mathrm{max}}$ this condition is not satisfied, then it is dropped and the algorithm requires only that one of Termination Tests 1, 2, or 3 is satisfied. In evaluating these tests, the second inequality of the Tangential Step Condition is enforced with its right-hand side set to zero, as in [10]. Since $\zeta\|v_{k,j,l}\|_2 \geq 0$, a step meeting this stronger requirement meets the condition as stated in §3.2 for every $\zeta \in \mathbb{R}_{>0}$, so no value of $\zeta$ need be specified. The constants $\epsilon_2$ and $\epsilon_3$ in Table 1 are those of Termination Tests 2 and 3 as stated in §3.2: $\epsilon_2$ determines when the dual-only step of TT2 may be considered, and $\epsilon_3$ is the fraction of the reduction in linearized infeasibility attained by the normal step that TT3 requires of the full step. (In the numbering of [10], the roles of these two constants are interchanged.) In addition, for a step to be accepted through TT2, our implementation requires the Residual Condition to hold with a constant $\kappa_2 \in (0, \kappa]$ in place of $\kappa$, so that a dual-only step is accepted only when the dual residual has been reduced sufficiently.

Two formulations of (11) are available in each iteration, and they differ as to whether a normal step is computed at all. The *direct* formulation replaces the lower right-hand-side block of (11) by $-c_{k,j,l}$, so that $J_{k,j,l}d_{k,j,l} = -c_{k,j,l}$, which yields $d_{k,j,l}$ in a single solve and requires no normal step. The *decomposed* formulation, by contrast, first computes $v_{k,j,l}$ by Algorithm 3, then solves (11) exactly as it is stated, thereby obtaining $d_{k,j,l} = v_{k,j,l} + u_{k,j,l}$. The direct formulation is the less expensive of the two, since it avoids the normal-step solve entirely. However, it offers no control over the tangential component when the iteration stalls: the Tangential Step Condition and the Model Reduction Condition are both stated in terms of $u_{k,j,l}$ and $v_{k,j,l}$ separately, and a short accepted step is the practical signal that these two quantities are in conflict. We therefore switch between the two formulations, following [10, §3.1]. Specifically, for a user-defined threshold $\kappa_{\mathrm{sw}} \in (0,1]$, an accepted step with $\alpha_{k,j,l} \geq \kappa_{\mathrm{sw}}$ selects the direct formulation for the subsequent iteration, whereas a shorter step selects the decomposed one, and each sampled problem begins with the direct formulation. Each Krylov solve starts from the zero vector.

Two further situations merit attention. First, the Hessian Modification Strategy is applied only once the same relative residual that (40) drives below $\kappa_{\mathrm{des}}$ has fallen to at most $\kappa_W$, since the curvature along an inaccurate iterate is not yet indicative of that of $W_{k,j,l}$ in the reduced space; see [10]. Whenever the strategy calls for a modification of $W_{k,j,l}$, the current Krylov solve is discarded and a new one, with its own allowance of $l^{\mathrm{pd}}_{\mathrm{max}}$ iterations, is begun from the modified matrix. Second, if an iterative solve ends without any of the three termination tests having been satisfied, whether by reaching $l^{\mathrm{pd}}_{\mathrm{max}}$ or through the safeguards above, then the trial step and all Hessian modifications accumulated along the way are discarded, and (11) is instead solved by the inertia-correcting factorization stated as Algorithm 4, which follows the default Ipopt algorithm [36], applied to the original $W_{k,j,l}$. A capped direct solve is first retried once with the decomposed formulation and a freshly computed normal step. The factorization applies only to the iteration in which it occurs; the next iteration begins with the iterative solver again. Because the dual regularization perturbs the constraint block, a direction obtained in this way need not satisfy any of Termination Tests 1, 2, or 3 for (11) as stated; it is accepted as a practical exception.

**Algorithm 4** Inertia Correction and Primal-Dual Solve

**Given** $W_{k,j,l}$, $J_{k,j,l}$, $g_{k,j,l}$, $y_{k,j,l}$, $v_{k,j,l}$, $\delta_w^0, \delta_c^0 \in \mathbb{R}_{>0}$
Set $\delta_w \leftarrow 0$, $\delta_c \leftarrow 0$, $a_w \leftarrow \delta_w^0$, and $a_c \leftarrow \delta_c^0$
**loop**
  Attempt to factor $K \leftarrow \begin{bmatrix} W_{k,j,l} + \delta_w I_{n+q} & J_{k,j,l}^T \\ J_{k,j,l} & -\delta_c I_{p+q} \end{bmatrix}$
  **if** the factorization fails **then**
    Set $\delta_w \leftarrow a_w$, $\delta_c \leftarrow a_c$, $a_w \leftarrow 10a_w$, and $a_c \leftarrow 10a_c$
  **else**
    Let $\varsigma_+$ and $\varsigma_-$ be the numbers of positive and negative eigenvalues of $K$, respectively
    **if** $\varsigma_+ = n + q$ **and** $\varsigma_- = p + q$ **then break**
    **if** $\varsigma_+ < n + q$ **then** $\delta_w \leftarrow a_w$ and $a_w \leftarrow 10a_w$
    **if** $\varsigma_- < p + q$ **then** $\delta_c \leftarrow a_c$ and $a_c \leftarrow 10a_c$
  **end if**
**end loop**
Solve $K \begin{bmatrix} d_{k,j,l} \\ \delta_{y,k,j,l} \end{bmatrix} = - \begin{bmatrix} g_{k,j,l} + J_{k,j,l}^T y_{k,j,l} \\ -J_{k,j,l} v_{k,j,l} \end{bmatrix}$ and **return** $(d_{k,j,l}, \delta_{y,k,j,l})$

### 5.6 Line search and iterate update

A few additional points should be mentioned about our implementation related to the line search and iterate update. First, rather than perform a standard penalty parameter update and backtracking line search with sufficient decrease condition as stated in (16), our implementation employs a flexible penalty function as described in [13]. Specifically, the implementation maintains lower and upper merit parameter values $\pi_{k,j,l}^l \leq \pi_{k,j,l}^u$, initialized to $\pi_0^l$ and $\pi_0^u$, where the upper value plays the role of $\pi_{k,j,l}$ in Algorithm 2; in particular, it is the value increased through (13). A step size $\alpha_{k,j,l} \in (0, \alpha_{k,j,l}^{\max}]$ is accepted if

$$\begin{aligned} &\phi_{\mathcal{S}_k}(x_{k,j,l} + \alpha_{k,j,l}\tilde{d}_{x,k,j,l}, s_{k,j,l} + \alpha_{k,j,l}\tilde{d}_{s,k,j,l}; \mu_{k,j}, \pi) \\ &\leq \phi_{\mathcal{S}_k}(x_{k,j,l}, s_{k,j,l}; \mu_{k,j}, \pi) - \eta_2 \alpha_{k,j,l} \Delta m_{k,j,l}(d_{k,j,l}; \mu_{k,j}, \pi_{k,j,l}^u) \end{aligned} \tag{41}$$

for either $\pi = \pi_{k,j,l}^l$ or $\pi = \pi_{k,j,l}^u$. If the accepted step satisfies (41) for $\pi = \pi_{k,j,l}^u$ but not for $\pi = \pi_{k,j,l}^l$, then the lower value is increased toward the value at which the two sides of (41) balance; namely, with $\varphi_{\mathcal{S}_k}(x, s; \mu) := f_{\mathcal{S}_k}(x) - \mu \sum_{i \in [q]} \ln(s^{(i)})$ and

$$\varrho_{k,j,l} := \frac{\varphi_{\mathcal{S}_k}(x_{k,j,l+1}, s_{k,j,l+1}; \mu_{k,j}) - \varphi_{\mathcal{S}_k}(x_{k,j,l}, s_{k,j,l}; \mu_{k,j})}{\|c_{k,j,l}\|_2 - \|c_{k,j,l+1}\|_2},$$

the implementation sets

$$\pi_{k,j,l+1}^l \leftarrow \min\left\{\pi_{k,j,l}^u,\ \pi_{k,j,l}^l + \max\{\gamma_l(\varrho_{k,j,l} - \pi_{k,j,l}^l),\ \epsilon_l\}\right\}$$

for user-defined $\gamma_l \in (0, 1)$ and $\epsilon_l \in \mathbb{R}_{>0}$; otherwise, it sets $\pi_{k,j,l+1}^l \leftarrow \pi_{k,j,l}^l$. Moreover, the line search is capped at $\bar{B}$ trials. If the line search does not terminate within this many trials, the final trial step is accepted rather than taking a null step. Second, in place of the minimization (19), our implementation takes $\beta_{y,k,j,l}$ to be the least value in $[\alpha_{k,j,l}, 1]$ satisfying the inequality required of $y_{k,j,l+1}$ in §3.2, namely, (18). Third, the tests for an infeasible stationary point in Algorithms 1 and 2 are implemented: A point $(x, s)$ is declared an infeasible stationary point when $\|c_{\mathcal{E}}(x)\|_\infty > \epsilon_{\text{infeas}}$ or $\|\max\{c_{\mathcal{I}}(x), 0\}\|_\infty > \epsilon_{\text{infeas}}$ while the residual of (4) is at most $\epsilon_{k,1}$ in the $\ell_\infty$-norm.

## 6 Numerical Experiments

Our aim in this section is to demonstrate the practical performance of our implementation of RA-IPM described in §5 through three sets of numerical experiments. The first tests the effect of the warm-starting strategy between sampled problems, the second demonstrates the improved performance of the RA-IPM

Table 1: Parameter values used for Algorithms 1, 2, 3, and 4.

| Symbol | Value | Description |
|---|---|---|
| *Sampling (§5.2)* | | |
| $N$ | 1024 | Full-sample size |
| $N_0$ | 128 | Initial sample size |
| $\tilde{\theta}$ | 0.5 | Sample-size accuracy parameter, (33) |
| $\hat{\beta}$ | 5 | Maximum sample-growth factor, (34) |
| *Warm-starting (§5.3)* | | |
| $\mu_0$ | $10^{-1}$ | Initial barrier parameter, Alg. 1 |
| $\bar{\rho}$ | 1 | Gradient-drift threshold, (35) and subsequent discussion |
| $\underline{\rho}$ | 1 | Drift denominator safeguard, (35) |
| $c_\mu$ | 2 | Barrier carry-forward factor, (37) |
| $c_s$ | $10^{-2}$ | Slack residual-budget coefficient, (38) |
| *Normal step (§5.4)* | | |
| $\omega$ | 100 | Initial trust-region scaling, (10) |
| $\omega_{\max}$ | $10^{20}$ | Maximum trust-region scaling |
| $\epsilon_v$ | 1 | Fraction of Cauchy decrease attained |
| $\delta_v$ | $10^{-8}$ | Initial normal-equation ridge |
| $\varsigma_v$ | 10 | Normal-equation ridge growth factor |
| *Primal-dual step (§5.5)* | | |
| $\kappa$ | $10^{-2}$ | Residual Condition constant |
| $\kappa_{\text{des}}$ | $10^{-3}$ | Desired relative residual tolerance, (40) |
| $\psi$ | 0.1 | Tangential Step Condition norm parameter |
| $\theta$ | $10^{-12}\mu_{k,j}$ | Tangential Step Condition curvature parameter |
| $\sigma$ | 0.09 | Model Reduction Condition constant |
| $\epsilon_2$ | $10^{-8}$ | Eligibility tolerance of the dual-only test (TT2) |
| $\epsilon_3$ | 0.9 | Feasibility-progress fraction of TT3 |
| $\kappa_2$ | $10^{-3}$ | Residual Condition constant for TT2, in place of $\kappa$ |
| $\kappa_W$ | $10^{-2}$ | Relative-residual threshold for Hessian modification, §5.5 |
| $\delta_P$ | $10^{-10}$ | Preconditioner diagonal floor and Schur-complement shift |
| $\tau_P$ | $10^{-3}$ | Preconditioner Schur drop tolerance |
| $\underline{\kappa}_{sz}$ | $10^{-10}$ | Lower safeguard on $s^{(i)}z^{(i)}/\mu_{k,j}$ |
| $\overline{\kappa}_{sz}$ | $10^{10}$ | Upper safeguard on $s^{(i)}z^{(i)}/\mu_{k,j}$ |
| $\kappa_{\text{sw}}$ | $10^{-3}$ | Direct/decomposed switching threshold |
| $\delta_w^0 = \delta_c^0$ | $10^{-5}$ | Initial inertia corrections, Alg. 4 |
| $\tilde{l}_{\max}^{\text{pd}}$ | 100 | Krylov iterations before (40) is dropped |
| $l_{\max}^{\text{pd}}$ | $\lceil 1.1(n+p+2q)\rceil$ | Krylov iteration limit per linear system, both phases |
| $\kappa_{\text{gap}}$ | 10 | Residual-discrepancy factor for leaving FGMRES |
| $\epsilon_{\text{gap}}$ | $10^{-14}$ | Absolute residual-discrepancy floor for leaving FGMRES |
| $l_{\text{stall}}^{\text{pd}}$ | 300 | Minimum iterative stagnation window |

Table 2: Parameter values used for Algorithms 1, 2, 3, and 4 (continued).

| Symbol | Value | Description |
|---|---|---|
| *Line search, barrier, and flexible penalty* | | |
| $\eta_1$ | $\max\{0.99,\, 1-\mu_{k,j}\}$ | Fraction-to-the-boundary parameter |
| $\eta_2$ | $10^{-8}$ | Sufficient-decrease constant, (16) |
| $\tau$ | 0.1 | Trial-penalty denominator factor, (13) |
| $\pi_0^l$ | $10^{-8}$ | Initial lower flexible-penalty value |
| $\pi_0^u$ | 1 | Initial upper flexible-penalty value |
| $\epsilon_\pi$ | $10^{-4}$ | Upper-penalty increment |
| $\gamma_l$ | $10^{-4}$ | Lower-penalty update fraction, §5.6 |
| $\epsilon_l$ | $10^{-4}$ | Minimum lower-penalty increment, §5.6 |
| $\epsilon_{\mu,a}$ | 10 | Barrier-termination factor, $a \in [5]$, (23) |
| $\kappa_\mu$ | 0.2 | Linear barrier-reduction factor, (36) |
| $\theta_\mu$ | 1.5 | Superlinear barrier-reduction exponent, (36) |
| $\underline{\mu}$ | $10^{-9}$ | Barrier-parameter floor, (36) |
| *Iteration limits* | | |
| $\bar{K}$ | 4 | Maximum outer RA iterations |
| $\bar{J}$ | 8 | Maximum barrier stages per sampled problem |
| $\bar{L}$ | 500 | Maximum inner iterations per barrier subproblem |
| $\bar{L}_{\text{det}}$ | 2000 | Inner-iteration limit in deterministic mode |
| $\bar{B}$ | 50 | Maximum line-search trials |
| *Termination tolerances, scaling, and test problems* | | |
| $\gamma_{k,a}$ | 0.5; 0 once $N_k = N$ | Relative RA tolerance, $a \in [5]$, (8) |
| $\epsilon_{k,a}$ | $10^{-5}$ | Absolute RA tolerance, $a \in [5]$, (8) |
| $\epsilon_{\text{full}}$ | $10^{-4}$ | Full-sample termination tolerance |
| $\epsilon_{\text{infeas}}$ | $10^{-4}$ | Infeasibility threshold for infeasible stationarity, §5.6 |
| $\Delta_\xi$ | $10^{-1}$ | Noise half-width, (42) |
| $g_{\max}$ | 100 | Gradient cap for problem scaling |
| $s_{\min}^{\text{scale}}$ | $10^{-8}$ | Minimum scaling factor, §5.1 |

algorithm due to adaptive sampling through retrospective approximation as compared with solving the full-sample problem directly, and the third shows the improvement in solution time that is possible due to the use of an interior-point method rather than an SQP method, specifically when solving larger-scale problems. Since the benefits of inexact search direction computations are only relevant for such larger-scale problems, in the first two experiments we employ a factorization-based approach for computing the search directions, and only employ the iterative linear system solves with inexact computations in the third experiment.

All of the results in this section were obtained on a MacBook Air with an Apple M2 processor, eight CPU cores, and 8 GB of memory running macOS 26.5.1. Our implementation is written in Python 3.11.11 with NumPy 2.4.2 [23] and SciPy 1.17.1 [35], the latter providing the sparse LU factorization used for the preconditioner of §5.5. Our test problems, described below, are accessed through PyCUTEst 1.8.0 [19] with CUTEst 2.6.0 [21]. The normal equations of Algorithm 3 and the symmetric indefinite systems of Algorithm 4 are factored with MUMPS 5.8.2 [1] through python-mumps 0.0.6. Our experiments also involve an implementation of RA-SQP [2], which uses Gurobi 13.0.2 [22] through gurobipy 13.0.2. We also show the results of runs using Ipopt as the subproblem solver; specifically, we employed Ipopt 3.14.19 [36], accessed through CyIpopt 1.6.1 with MUMPS as its linear solver.

The parameters of Algorithms 1 and 2 were set as stated in Tables 1–2, except where an experiment states otherwise. Ipopt departs from its defaults in that its own scaling is disabled in favor of the scaling we implemented as described in the beginning of §5. We also set its iteration limit to match that of RA-IPM and modified its stopping test through its iteration callback so that it terminates on the conditions of (8). The remaining settings were left at Ipopt defaults.

We generated our test set from the CUTEst collection [21] through the PyCUTEst interface [19], proceeding as follows. Firstly, we retained the problems whose constraints are classified as quadratic or more general, and removed all others. Secondly, we expressed each such problem in the form (1) and removed those possessing no inequality constraint that is nonlinear since RA-IPM is primarily designed for problems with at least one inequality constraint that is nonlinear. Thirdly, we removed those problems with $n+p+q > 7000$. Finally, we removed those problems whose objective, constraint, or derivative values are not finite at the prescribed starting point, namely LHAIFAM, S365, and S365MOD. Our test set was the remainder of these problems, which resulted in 169 total problems.

The experiments of §6.1 and §6.2 measure component evaluations, where each evaluation of $\nabla f_{\mathcal{S}_k}$ during these experiments costs $N_k$ gradient component evaluations. The experiment in §6.3, on the other hand, measures CPU time. For the latter, the dominant cost is that of solving (11) in each iteration. As an estimator for that cost, we considered the number of nonzeros (nnz) of the matrix. We accordingly measured the size of a problem $P$ by the number of nonzeros in

$$K_0(P) := \begin{bmatrix} W_0 & J_0^T \\ J_0 & 0 \end{bmatrix} \in \mathbb{R}^{(n+p+2q)\times(n+p+2q)},$$

where $W_0$ and $J_0$ are the matrices of §3.2 evaluated at the initial primal-dual iterate of the scaled problem with barrier parameter $\mu_0$. Let us refer to this count as $\mathrm{nnz}(K_0(P))$. We defined a *small* set $\mathcal{P}_\mathrm{s}$ of problems, employed in the first two experiments, consisting of the 126 (out of 169) problems with $\mathrm{nnz}(K_0(P)) \leq 2000$, whereas we defined a *large* set $\mathcal{P}_\mathrm{l}$ of problems consisting of the remaining 43 problems, which we employed in the third experiment.

Following [2], we modified the default CUTEst problems with an additional objective term in order to allow for sampling through retrospective approximation. Letting $f_\mathrm{orig}$ denote the original CUTEst objective function, $x_\mathrm{init}$ denote the prescribed starting point, and $e_n \in \mathbb{R}^n$ denote a vector of ones, each objective is modified to be

$$f(x) = f_\mathrm{orig}(x) + \frac{1}{N}\sum_{i=1}^{N} \xi_i \|x - x_\mathrm{init} - e_n\|_2^2, \tag{42}$$

where $\xi_i$ for each $i \in [N]$ is drawn from a uniform distribution over $[-\Delta_\xi, \Delta_\xi]$. Each sampled objective $f_\mathcal{S}$ is formed by choosing uniformly, at random, from $\{\xi_i\}_{i\in[N]}$ without replacement. However, since such a selection might lead to $\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \xi_i$ being negative, a selection is thrown out and re-drawn to ensure that

$\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\xi_i > 0$. The same goes for the full-sample problem to ensure that $\frac{1}{N}\sum_{i=1}^{N}\xi_i > 0$. Each method that was employed to solve a given problem received the same seed, which ensured that, when comparing different methods for solving the same problem, the full sample $\mathcal{S}_{\text{full}}$ was the same across methods.

Finally, let us state how we measure performance. A run is successful when, at some iterate, the residuals on the left-hand side of (8), evaluated with the full-sample gradient, satisfy $\max_{a\in[5]}\|r_a\|_\infty \le \epsilon_{\text{full}} := 10^{-4}$. All five residuals are employed for runs employing RA-IPM and Ipopt. RA-SQP, on the other hand, maintains no multipliers for the bounds on the slack variables, so for it the criterion is applied with residuals $r_1$, $r_3$, $r_4$, and the norm of $Sy_{\mathcal{I}}$. Runs that exhaust an iteration limit, that stop at an infeasible stationary point, or that fail numerically are counted as unsuccessful. We report the performance using Dolan–Moré performance profiles [17]. Thus, for each problem the measure of a method is divided by the smallest measure attained on that problem by any method under comparison, where an unsuccessful run is assigned an infinite ratio. Each method is run once per problem and configuration.

## 6.1 Warm-starting between sampled problems

Our first set of experiments examines the warm-starting strategy of §5.3. We compare the *warm* configuration, which applies (37) and (39), against a *cold* configuration, which in every outer iteration sets $\mu_{k,0} = \mu_0$, restores the full budget of $\bar{J}$ barrier stages, and sets $\iota_{s,k} = 0$ and $\iota_{z,k} = 0$. In both configurations the primal-dual iterate returned by each inner solve is carried forward, as line 9 of Algorithm 1 prescribes, so the two runs differ only in the barrier parameter, the barrier-stage budget, and the slack variables with which each inner solve begins. Both are run over $\mathcal{P}_{\text{s}}$ with the same parameter values. We report profiles of function component evaluations, gradient component evaluations, and inner iterations.

Figure 1 provides the results. The warm configuration solves 121 of the 126 problems and the cold configuration 120, all of which are also solved by the warm configuration. For all three measures the warm profile starts above the cold one at a performance ratio of one, but the size of the effect differs. It is clearest for gradient component evaluations: on 115 of the 120 problems solved by both configurations, the warm configuration uses fewer of them. For function component evaluations and inner iterations, the cold profiles catch up within a small factor.

This pattern reflects the way in which the cold configuration restarts. Its first barrier subproblem in each outer iteration uses $\mu_0$, whereas the iterate carried forward is nearly stationary for the previous sampled problem. The barrier termination test (23), with $\epsilon_{\mu,a} = 10$, holds at such a point for every sufficiently large value of the barrier parameter, so several barrier stages terminate at their entry test without taking a step. Each such stage costs a sampled-gradient evaluation for its entry test but records no inner iteration. Warm-starting therefore saves gradient evaluations mainly by sparing these stages, and saves comparatively few inner iterations. The slack perturbation (39) is small by construction, with $\|\iota_{s,k}\|_\infty \le c_s\epsilon_{k,4} = 10^{-7}$ for the values in Tables 1–2, and the experiment does not separate its effect from that of the barrier parameter.

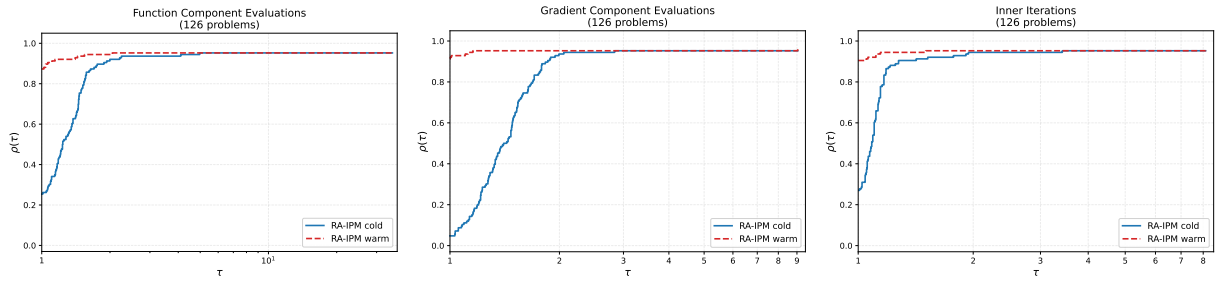



Figure 1: Performance profiles for the warm and cold configurations of RA-IPM on the 126 problems of $\mathcal{P}_{\text{s}}$: function component evaluations, gradient component evaluations, and inner iterations.

## 6.2 Adaptive sampling against deterministic solves

The paradigm of RA rests upon the premise that the early outer iterations do not require accurate objective information. Should this premise hold, a method that begins with $N_0 = 128$ realizations and enlarges the sample only as the residual becomes small ought to reach the target accuracy with fewer component evaluations than one that employs all $N$ realizations from the start. The premise is not automatic, however, since enlarging the sample alters the problem being solved, so that part of the work performed over $\mathcal{S}_{k-1}$ is discarded. Our second set of experiments tests this premise, and does so for three inner solvers, so that any advantage observed may be attributed to the outer loop rather than to the subproblem solver.

We compare the *adaptive* mode of §5.2 against a *deterministic* mode in which the outer loop receives $\mathcal{S}_{\text{full}}$ in its first iteration. In deterministic mode, one has $N_0 = N$, so that by the rule of §5.2 the relative tolerances $\gamma_{k,a}$ vanish and only the absolute conditions in (8) remain. In adaptive mode, the relative tolerances are likewise set to zero once the sample reaches $\mathcal{S}_{\text{full}}$. In both modes, each method also evaluates the residuals with the full-sample gradient in every inner iteration and stops as soon as the success criterion stated at the beginning of this section holds. These full-sample evaluations serve only to detect success and are not counted. That said, all other evaluations are counted, including those used to estimate the variance in (32). The measures reported below are therefore the costs of reaching the target accuracy $\epsilon_{\text{full}}$, which in adaptive mode may be reached before the sample has grown to $\mathcal{S}_{\text{full}}$. We note that this stopping rule relies on the full-sample gradient, which would not be available in the setting of §2. In that setting, the method would instead continue until $N_k = N$ and the absolute tolerances $\{\epsilon_{k,a}\}$ are met. We employ the rule, identically in both modes, so that the two modes are compared on the cost of attaining the same accuracy for the same instance of (1). Measured instead at each mode's own termination, the adaptive mode would also be charged for the full-sample solve that follows.

The three inner solvers are RA-IPM, RA-SQP [2], and Ipopt [36]. Ipopt is driven through its iteration callback and terminated by the conditions (8), as RA-IPM is; at the start of each sampled problem it is started from the current primal iterate with initial barrier parameter $\mu_0$ and its default initialization of the multipliers. RA-SQP terminates each sampled solve by its step-norm test [2]. In each case the inner solver is applied to the sampled problem within Algorithm 1 and receives the same $\mathcal{S}_{\text{full}}$ in both modes, so that each pair of profiles constitutes a comparison within a single inner solver. In particular, the deterministic runs are not stock Ipopt or stock RA-SQP applied to a different problem, and we draw no conclusion across the three pairs. The iteration limits are $\bar{L} = 500$ in adaptive mode and $\bar{L}_{\text{det}} = 2000$ in deterministic mode for both RA-IPM and RA-SQP, and 3000 for Ipopt in both modes. The units differ: for RA-IPM the limit applies to each barrier subproblem, whereas for RA-SQP and Ipopt it applies to each sampled solve.

We report function and gradient component evaluations, as well as total inner iterations. A component evaluation is an evaluation of $F(\cdot, \xi)$, or of its gradient, for one realization $\xi$, so that an evaluation of $f_{\mathcal{S}_k}$ or of $\nabla f_{\mathcal{S}_k}$ is charged $N_k$ components. One expects adaptive sampling to prevail on the first two measures, since its early evaluations involve only a fraction of the full sample, and to fare worse on the third, since it may solve several problems rather than one. For RA-IPM and Ipopt the third measure counts inner solver iterations, whereas for RA-SQP it counts the iterations consumed by Gurobi within the subproblem solves.

Figures 2–4 provide the results. For each of the three inner solvers, the adaptive profiles for component evaluations start well above the deterministic ones at a performance ratio of one, whereas the deterministic profiles approach their final levels only at larger ratios. With RA-IPM the adaptive profile is flat from a ratio of one, so the adaptive mode attains the smaller count on essentially every problem it solves; with RA-SQP and Ipopt the advantage is less uniform. The inner-iteration profiles, by contrast, favor the deterministic mode for each pair, so the savings come from the cost of each evaluation rather than from the number of iterations, which is the pattern that the RA paradigm predicts. The two modes are similarly reliable, the adaptive mode solving two problems more than the deterministic mode with RA-IPM, one fewer with RA-SQP, and two fewer with Ipopt.

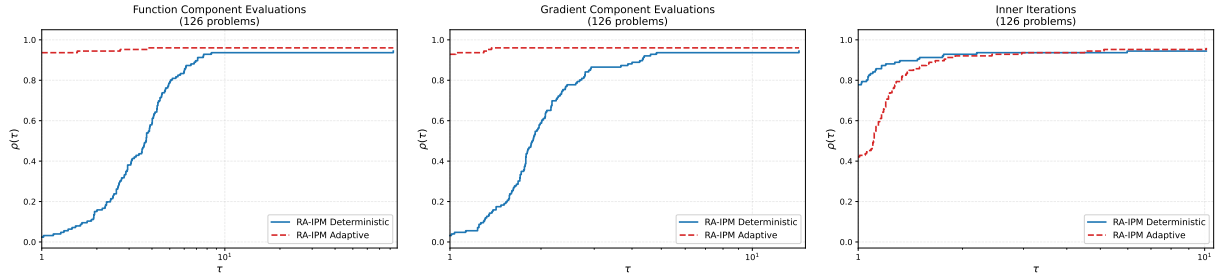



Figure 2: Performance profiles for the deterministic and adaptive modes with RA-IPM on the 126 problems of $\mathcal{P}_s$: function component evaluations, gradient component evaluations, and inner iterations.

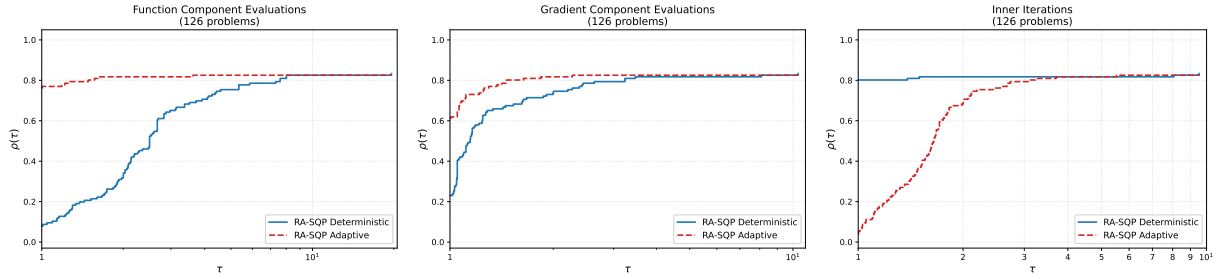



Figure 3: Performance profiles for the deterministic and adaptive modes with RA-SQP on the 126 problems of $\mathcal{P}_s$: function component evaluations, gradient component evaluations, and inner iterations, here counted as Gurobi subproblem iterations.

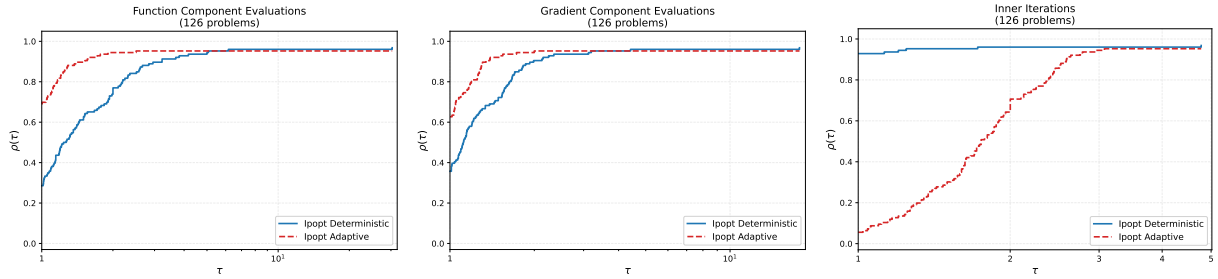



Figure 4: Performance profiles for the deterministic and adaptive modes with Ipopt on the 126 problems of $\mathcal{P}_s$: function component evaluations, gradient component evaluations, and inner iterations.

### 6.3 Problems with larger KKT systems

Over our set of larger problems, the matrix in (11) has more than 2000 and at most 4,007,000 nonzeros at the starting point. These problems are not of small size, so this experiment concerns the regime in which the linear algebra of a step begins to dominate the cost of a run. Our set of experiments here compares RA-IPM against RA-SQP [2] on these problems. For deterministic problems, the advantage of interior-point methods over SQP methods as problems grow is generally recognized. The question here is whether this advantage persists when the solver must be restarted for each new sampled problem.

The two methods compute a step quite differently. An RA-IPM iteration requires an approximate solution of one linear system of the form (11), obtained by the Krylov procedure of §5.5, with a fallback to the inertia-correcting factorization of Algorithm 4 when no candidate direction satisfies a termination test. Such an iteration still involves sparse factorizations, namely of the approximate Schur complement used by the preconditioner and, in the decomposed formulation, of the matrix of the normal equations in Algorithm 3, but the matrix in (11) itself is factored only when the fallback is invoked. An RA-SQP iteration, by contrast, solves a dual linear program, a feasibility linear program, and a quadratic program with Gurobi, and computes a smallest-eigenvalue estimate for the QP Hessian matrix. For our experiments here, both methods are run with the same $\mathcal{S}_{\text{full}}$, the same seed, and the same sampling rule, although the samples they draw need not agree since the sample sizes depend on the iterates through the rule of §5.2.

Over $\mathcal{P}_{\text{l}}$ the time consumed by a run is dominated by the linear algebra of the step computation rather than by the evaluation of $f_{\mathcal{S}_k}$ and $\nabla f_{\mathcal{S}_k}$, so component evaluation counts no longer discriminates between the methods. Iteration counts are not comparable either, since the two methods perform very different work per iteration. We therefore compare CPU time, measured as the processor time of each run summed over all of its threads. NumPy, SciPy, and MUMPS were restricted to a single thread, Gurobi was run with its default settings, and no time limit was imposed. CPU time compares implementations as well as algorithms. RA-SQP solves its subproblems with Gurobi, a mature commercial code, but each of its iterations also incurs the cost of building those subproblems and of the eigenvalue estimate. RA-IPM, on the other hand, is research code and uses publicly available linear algebra routines. Nonetheless, the comparison should be read as indicating the relative cost of the two approaches as implemented here, rather than as a benchmark of solvers.

Figure 5 reports results for 41 of the 43 problems of $\mathcal{P}_{\text{l}}$. (On the remaining two, the runs of both methods exhausted the available memory, so we omit them.) RA-IPM solves 35 of the 41 problems while RA-SQP only solves 16. Fifteen problems are solved by both methods, 20 only by RA-IPM, and one only by RA-SQP. RA-IPM attains the smallest CPU time on 34 of the 41 problems and RA-SQP on 2. This ordering reflects an advantage in time and not only in reliability, since on the 15 problems solved by both methods, RA-IPM is the faster on 14. On this test set and with our implementations, RA-IPM was thus both more reliable and faster than RA-SQP.

## 7 Conclusion

We have proposed RA-IPM, an algorithm for minimizing an expectation subject to deterministic nonlinear equality and inequality constraints. Its outer loop follows the retrospective-approximation paradigm, solving a sequence of sample-average problems with increasing sample sizes, and each sampled problem is solved approximately by an interior-point method that allows inexact step computations. The inner method stops on the same conditions under which the outer loop is analyzed, namely a relative reduction, up to an absolute tolerance, of each block of the KKT residual.

Under our assumptions, if the algorithm generates an infinite sequence of iterates, the sample sizes grow without bound, and the tolerances vanish, then the expected KKT residuals at the outer iterates converge to zero. These assumptions include a bound on the sampling error of the gradient at the points returned by the inner solves. The analysis is asymptotic and does not bound the total work of the method.

In experiments on CUTEst problems whose objectives were augmented with a sampled term, carrying the barrier parameter and the slack variables from one sampled problem to the next reduced the number

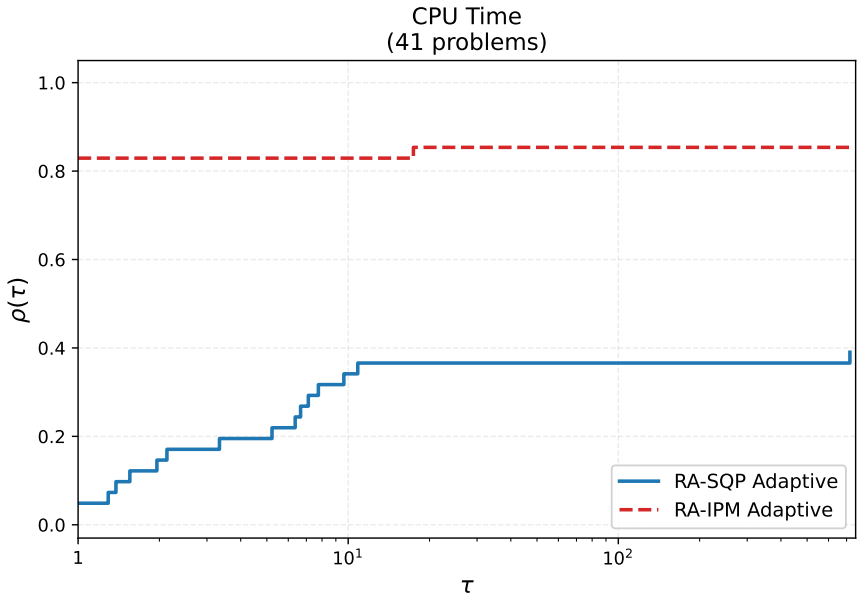



Figure 5: Performance profile of CPU time for RA-SQP and RA-IPM, both in adaptive-sampling mode, on 41 of the 43 problems of $\mathcal{P}_1$; see the text for the remaining two.

of gradient component evaluations relative to restarting the barrier parameter, mainly by sparing barrier stages. For each of three inner solvers, solving a sequence of sampled problems of increasing size reduced the number of component evaluations needed to reach the target accuracy on the full-sample problem, at the price of more inner iterations. On problems with larger KKT systems, RA-IPM solved more problems than RA-SQP and was faster on all but one of the problems that both solved.

# References


[1] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] Albert S. Berahas, Raghu Bollapragada, and Shagun Gupta. Retrospective approximation sequential quadratic programming for stochastic optimization with general deterministic nonlinear constraints. arXiv:2505.19382, 2025.

[3] Albert S. Berahas, Raghu Bollapragada, and Baoyu Zhou. An adaptive sampling sequential quadratic programming method for equality constrained stochastic optimization. arXiv:2206.00712, 2022.

[4] Albert S. Berahas, Frank E. Curtis, Michael J. O'Neill, and Daniel P. Robinson. A stochastic sequential quadratic optimization algorithm for nonlinear-equality-constrained optimization with rank-deficient Jacobians. *Mathematics of Operations Research*, 49(4):2212–2248, 2024.

[5] Albert S. Berahas, Frank E. Curtis, Daniel P. Robinson, and Baoyu Zhou. Sequential quadratic optimization for nonlinear equality constrained stochastic optimization. *SIAM Journal on Optimization*, 31(2):1352–1379, 2021.

[6] Raghu Bollapragada, Richard Byrd, and Jorge Nocedal. Adaptive sampling strategies for stochastic optimization. *SIAM Journal on Optimization*, 28(4):3312–3343, 2018.

[7] Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching L-BFGS method for machine learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 620–629. PMLR, 2018.

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

[9] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[10] Frank E. Curtis, Johannes Huber, Olaf Schenk, and Andreas Wächter. A note on the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations. *Mathematical Programming, Series B*, 136(1):209–227, 2012.

[11] Frank E. Curtis, Xin Jiang, and Qi Wang. A single-loop stochastic feasible interior-point algorithm for nonlinear inequality-constrained optimization. *Mathematical Programming*, pages 1–38, 2026.

[12] Frank E. Curtis, Vyacheslav Kungurtsev, Daniel P. Robinson, and Qi Wang. A stochastic-gradient-based interior-point algorithm for solving smooth bound-constrained optimization problems. *SIAM Journal on Optimization*, 35(2):1030–1059, 2025.

[13] Frank E. Curtis and Jorge Nocedal. Flexible penalty functions for nonlinear constrained optimization. *IMA Journal of Numerical Analysis*, 28(4):749–769, 2008.

[14] Frank E. Curtis, Daniel P. Robinson, and Baoyu Zhou. Sequential quadratic optimization for stochastic optimization with deterministic nonlinear inequality and equality constraints. *SIAM Journal on Optimization*, 34(4):3592–3622, 2024.

[15] Frank E. Curtis, Daniel P. Robinson, and Baoyu Zhou. A stochastic inexact sequential quadratic optimization algorithm for nonlinear equality-constrained optimization. *INFORMS Journal on Optimization*, 6(3–4):173–195, 2024.

[16] Frank E. Curtis, Olaf Schenk, and Andreas Wächter. An interior-point algorithm for large-scale nonlinear optimization with inexact step computations. *SIAM Journal on Scientific Computing*, 32(6):3447–3475, 2010.

[17] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[18] Yuchen Fang, Jihun Kim, Sen Na, James Demmel, and Javad Lavaei. A trust-region interior-point stochastic sequential quadratic programming method. arXiv:2603.10230, 2026.

[19] Jaroslav Fowkes, Lindon Roberts, and Árpád Bürmen. PyCUTEst: An open source Python package of optimization test problems. *Journal of Open Source Software*, 7(78):4377, 2022.

[20] Soumyadip Ghosh and Raghu Pasupathy. On interior-point based retrospective approximation methods for solving two-stage stochastic linear programs. In *Proceedings of the 2011 Winter Simulation Conference*, pages 4163–4171, 2011.

[21] Nicholas I. M. Gould, Dominique Orban, and Philippe L. Toint. CUTEst: A constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[22] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. `https://www.gurobi.com`, 2025.

[23] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[24] Carsten Keller, Nicholas I. M. Gould, and Andrew J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[25] O. L. Mangasarian and S. Fromovitz. The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, 17:37–47, 1967.

[26] Sen Na, Mihai Anitescu, and Mladen Kolar. Inequality constrained stochastic nonlinear optimization via active-set sequential quadratic programming. *Mathematical Programming*, 202(1):279–353, 2023.

[27] David Newton, Raghu Bollapragada, Raghu Pasupathy, and Nung Kwan Yip. A retrospective approximation approach for smooth stochastic optimization. *Mathematics of Operations Research*, 50(3):2301–2332, 2025.

[28] Raghu Pasupathy. An introspective on the retrospective-approximation paradigm. In *Proceedings of the 2011 Winter Simulation Conference*, pages 412–421, 2011.

[29] Johannes O. Royset. Optimality functions in stochastic programming. *Mathematical Programming*, 135:293–321, 2012.

[30] Johannes O. Royset. On sample size control in sample average approximations for solving smooth stochastic programs. *Computational Optimization and Applications*, 55(2):265–309, 2013.

[31] Johannes O. Royset and Elijah Polak. Implementable algorithm for stochastic optimization using sample average approximations. *Journal of Optimization Theory and Applications*, 122(1):157–184, 2004.

[32] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[33] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[34] Robert J. Vanderbei and David F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[35] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

[36] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.