# Fast Adaptive First-Order Semidefinite Programming for Data-Driven Linear Quadratic Regulation

Baghbadorani, Reza Rahimi; Esfahani, Peyman Mohajerin; Grammatico, Sergio

**DOI**
10.1109/MED61351.2024.10566178

**Publication date**
2024

**Document Version**
Final published version

**Published in**
Proceedings of the 32nd Mediterranean Conference on Control and Automation, MED 2024

2024 32nd Mediterranean Conference on Control and Automation (MED) | 979-8-3503-9544-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/MED61351.2024.10566178

# Fast Adaptive First-Order Semidefinite Programming for Data-Driven Linear Quadratic Regulation

Reza Rahimi Baghbadorani, Peyman Mohajerin Esfahani, and Sergio Grammatico

*Abstract*— **We study the data-driven finite-horizon linear quadratic regularization (LQR) problem reformulated as a semidefinite program (SDP). Our contribution is to propose two novel accelerated first-order methods for solving the resulting SDP. Our methods enjoy adaptive stepsize and adaptive smoothing parameters that speed up convergence and in turn, enhance scalability. Finally, we compare our accelerated first-order methods and show their benefits via numerical simulations on a benchmark LQR example.**

## I. INTRODUCTION

Linear matrix inequalities and in general semidefinite programs (SDP) are ubiquitous in system analysis and control design [1]. Therefore, semidefinite programming has become an important tool especially in data-driven control design because both the decision variables and the constraints are high-dimensional. In fact, data-driven control requires a large number of sampled data, which in turn, increases the number of decision variables in the associated SDP [2], [3].

For solving an SDP with high precision, second-order methods, such as the interior point method, are available, but suffer from high computational complexity [4], [5], [6]. Instead, first-order methods are the most popular for the high-dimensional SDPs. Specifically, two classes of first-order algorithms are mainly used: subgradient [7] and mirror descent [8] (which exploits the structure of the constraints). In their original formulation, these methods require $\mathcal{O}\left(1/\varepsilon^2\right)$ iterations to reach an $\varepsilon$-neighbor of an optimal solution.

Thanks to the pioneering work by Nesterov [9], [10], one can exploit the structure of the objective function in the SDP, use its soft-max smooth approximation, and then apply an accelerated first-order method [11]. Remarkably, Nesterov's algorithm needs $\mathcal{O}\left(1/\varepsilon\right)$ iterations to reach an optimal solution with $\varepsilon$ precision.
The main practical problem with [9] and [10] is perhaps that one should fix the smoothing parameter based on the desired precision, which however affects the stepsize choice. Consequently, having a small error $\varepsilon$ requires a small stepsize, which in turn reduces the convergence speed. Some recent works attempt to address this issue. The authors in [12] use an adaptive smoothing parameter with a conditional gradient descent with convergence rate $\mathcal{O}\left(1/\varepsilon^2\right)$. Another adaptive smoothing technique is proposed in [13] with $\mathcal{O}\left(1/\varepsilon\right)$ iteration complexity. The technical issue with this method is that the stepsize and the smoothing parameter decrease strictly throughout the iterations, thus resulting in slower convergence in practice.

In this paper, based on Nesterov's smoothing technique, we propose an adaptive stepsize-adaptive smoothing technique for SDPs that we apply to the data-driven finite-horizon LQR problem [14], [15]. Our theoretical contribution is to prove the convergence rate $\mathcal{O}\left(1/\varepsilon\right)$ for our proposed algorithms. Thanks to the adaptive stepsize and adaptive smoothing parameters we obtain faster convergence in practice. Differently from [13], the stepsize is non-increasing and in fact, in our numerical experience, it does not strictly decrease in many iterations. Instead, our smoothing parameter strictly decreases in each iteration, which results in a smaller approximation error. In summary, our adaptive (non-increasing) stepsize yields the best theoretical limit convergence rate ($\mathcal{O}\left(1/\varepsilon\right)$), while the adaptive smoothing parameter enhances the precision of the algorithm itself.

Next, when applying our methods to the data-driven finite-horizon LQR, we make the same (surprising) observation as in [16]: By increasing the number of samples in data-driven finite-horizon LQR, the number of iterations needed to reach the desired precision is decreased. One plausible explanation is that the information on the system dynamics is richer with more sampled data and thus the task of the solver is simplified. It is worth noting that by increasing the number of samples, the complexity per iteration grows. Nevertheless, we do not notice any noteworthy changes in our numerical experiments.

The paper is organized as follows. Section II contains the problem statement and rewrites the LQR problem as a semidefinite program; in Section III we propose our novel first-order methods for solving SDPs; Section IV benchmarks our algorithm in a data-driven finite-horizon LQR design. We conclude the paper in Section V where we highlight some future research directions.

**Notation**: For a square matrix $A$, $\mathbf{Tr}(A)$ refers to its trace. $A \succcurlyeq 0$ denotes that matrix $A$ is positive semidefinite. We use the notation $w(k) \sim \mathcal{N}(0, W)$ to show the Normal

R. Rahimi Baghbadorani, P. Mohajerin Esfahani, and S. Grammatico are with the Delft Center for Systems and Control, TU Delft, The Netherlands. E-mail addresses: {R.Rahimibaghbadorani, P.MohajerinEsfahani, S.Grammatico}@tudelft.nl.
This work was supported by the ERC under the research project TRUST-91562 and COSMOS-91409.

distribution with zero mean and variance $W$. The notation $B = \operatorname{diag}(A_1, \dots, A_n)$ is utilized to denote the block diagonal matrix $B$ with sub-matrices $A_1, \dots, A_n$ as diagonal components.

## II. LINEAR QUADRATIC REGULATION VIA SEMIDEFINITE PROGRAMMING

### A. Model-based linear quadratic regulation

Consider a discrete-time linear system

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, and $A$ and $B$ are the dynamics matrices of (1). The states of the system are assumed to be measurable and the pair $(A, B)$ is controllable. To compute the optimal control sequence $u_{[0,N-1]} = (u(0), \dots, u(N-1))$ over the horizon $N \in \mathbb{N}$ one might solve the following stochastic linear quadratic control problem [17]:

$$\begin{cases} \min\limits_{\rho_k} & \mathbf{E} x^\top(N) Q_{\text{xx}} x(N) + \\ & \sum_{k=0}^{N-1} \mathbf{E} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^\top Q \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \\ \text{s.t.} & x(k+1) = Ax(k) + Bu(k) + w(k) \\ & x(0) \sim \mathcal{N}(0, I_n) \\ & w(k) \sim \mathcal{N}(0, I_n) \\ & \mathbf{E}[w(k) x^\top(l)] = 0, \quad \forall l \le k \\ & u(k) = \rho_k(x(0), x(1), \dots, x(k)) \end{cases} \tag{2}$$

where

$$Q = \begin{bmatrix} Q_{\text{xx}} & 0 \\ 0 & Q_{\text{uu}} \end{bmatrix} > 0.$$

Moreover, according to [17, Prop. 1], (2) can be reformulated as the following covariance selection problem:

$$\begin{cases} \min\limits_{V(0),\dots,V(N) \succcurlyeq 0} \mathbf{Tr}\left(Q_{\text{xx}} V_{\text{xx}}(N)\right) \\ \qquad\qquad + \sum_{k=0}^{N-1} \mathbf{Tr}(Q V(k)) \\ \text{s.t.} \\ V_{\text{xx}}(0) = I_n \\ \begin{bmatrix} A & B \end{bmatrix} V(k) \begin{bmatrix} A & B \end{bmatrix}^\top + I_n = V_{\text{xx}}(k+1) \\ \forall k \in \{0, \dots, N-1\} \end{cases} \tag{3}$$

where

$$V(k) = \begin{bmatrix} V_{\text{xx}} & V_{\text{xu}} \\ V_{\text{xu}}^\top & V_{\text{uu}} \end{bmatrix} = \mathbf{E} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}^\top,$$

and the corresponding control law is then

$$u(k) = V_{\text{xu}}^\top(k) V_{\text{xx}}^{-1} x(k). \tag{4}$$

In view of [14, Th. 2] we can reformulate the covariance selection problem in (3) as an SDP which can be solved via first-order optimization algorithms.

**Lemma 1 (LQR via SDP)** [14, Th. 2] *The optimal control law in* (4) *is equivalent to the $\mathcal{K}$ solution of the following problem:*

$$\begin{aligned} \min_{\mathcal{V}_{\text{xx}}, \mathcal{K}, \mathcal{Z}} \; & \mathbf{Tr}(Q_{\text{xx}} V_{\text{xx}}(N)) \\ & + \sum_{k=0}^{N-1} \mathbf{Tr}(Q_{\text{xx}} V_{\text{xx}}(k) + Q_{\text{uu}} Z(k)) \\ \text{s.t.} \; & \\ & V_{\text{xx}}(0) \succcurlyeq I_n \\ & V_{\text{xx}}(k+1) - (A + BK(k)) V_{\text{xx}}(k) (A + BK(k))^\top \\ & \quad - I_n \succcurlyeq 0 \\ & Z(k) - Q_{\text{uu}}^{1/2} K(k) V_{\text{xx}}(k) K(k)^\top Q_{\text{uu}}^{1/2} \succcurlyeq 0 \end{aligned} \tag{5}$$

$\forall k \in \{0, \dots, N-1\}$, *where*

$$\begin{aligned} \mathcal{V}_{\text{xx}} &:= \{V_{\text{xx}}(1), \dots, V_{\text{xx}}(N)\} \\ \mathcal{K} &:= \{K(0), \dots, K(N-1)\} \\ \mathcal{Z} &:= \{Z(0), \dots, Z(N-1)\}. \end{aligned}$$

■

With the aim of applying our own optimization algorithms, let us first rewrite (5) in standard SDP form. Thus, we define an auxiliary variable $H(k) = K(k) V_{\text{xx}}(k)$, and exploit the fact that $V_{\text{xx}} \succcurlyeq I_n$ and the Schur complement inequality. Next, we define the matrices:

$$C(k) = \begin{bmatrix} V_{\text{xx}}(k+1) - I_n & AV_{\text{xx}}(k) + BH(k) \\ (AV_{\text{xx}}(k) + BH(k))^\top & V_{\text{xx}}(k) \end{bmatrix}_{2n \times 2n}$$

$$D(k) = \begin{bmatrix} Z(k) & Q_{\text{uu}}^{1/2} H(k) \\ (Q_{\text{uu}}^{1/2} H(k))^\top & V_{\text{xx}}(k) \end{bmatrix}_{(m+n) \times (m+n)}$$

and the block diagonal matrices

$$\begin{aligned} X &= \operatorname{diag}\left(C(1), D(1), \dots, C(n), D(n)\right) \\ F &= \operatorname{diag}\left(0_{2n \times 2n}, Q_{\text{uu}}, Q_{\text{xx}}, \dots, 0_{2n \times 2n}, Q_{\text{uu}}, Q_{\text{xx}}\right) \end{aligned}$$

Then (5) is equivalent to

$$\begin{cases} \min\limits_{X} \; \mathbf{Tr}(F^\top X) \\ \text{subject to } X \in \mathbb{S}_+^{N(3n+m) \times N(3n+m)} \end{cases} \tag{6}$$

The sparse structure of $X$ allows us to simplify the SDP in (6). In fact, thanks to the symmetry of $V_{\text{xx}}$, $Z$, $C$, and $D$, we have $\alpha = N(\frac{n}{2}(5n+3) + mn + \frac{m}{2}(m+1))$ variables. Therefore, we can write (6) in vector form as

$$\begin{cases} \min\limits_{y \in \mathbb{R}^\alpha} \; c^\top y \\ \qquad \text{s.t.} \quad G_0 + \mathcal{G}^*(y) \succcurlyeq 0 \\ \qquad\qquad\;\; Py = b \end{cases} \tag{7}$$

where $c \in \mathbb{R}^\alpha$, $\mathcal{G}^*(y) = \sum_{i=1}^{\alpha} y_i G_i$, $G_i, G_0 \in \mathbb{S}^{N(3n+m) \times N(3n+m)}$, $P \in \mathbb{R}^{N(2n^2+n) \times \alpha}$, and $b \in \mathbb{R}^{N(2n^2+n)}$ can be written explicitly by $A$, $B$, $Q_{\text{xx}}$, $Q_{\text{uu}}$, and $F$. Finally, the problem in (7) is in standard form of dual semidefinite programming [5]. We note that SDP in (7) requires the knowledge of the dynamics in (1).

### B. Data-driven linear quadratic regulation

A recently popular alternative to model knowledge is represented by the so-called data-driven approach where input-state experimental data are available [18], [19]. Let us consider the data-driven finite horizon LQR problem via SDP as well.

**Lemma 2 (Data-driven LQR via SDP)** [14, Th. 3] *Consider sampled input $d_{u,[0,T]}$ and state $d_{x,[0,T]}$ data with length $T$ of system* (1). *Let matrices $U_{0,T}$, $X_{0,T}$, and $X_{1,T}$ be the Hankel matrices of the experiment data satisfying the persistent excition condition. Then the optimal control law for system* (1) *is given by*

$$K(k) = U_{0,T} R(k) V_{\mathrm{xx}}^{-1}(k)$$

*where the matrices $R(k)$ and $V_{\mathrm{xx}}(k)$ are given by solving the following optimization problem:*

$$\begin{aligned}
\min_{\mathcal{V}_{\mathrm{xx}},\mathcal{R},\mathcal{Z}} \quad & \mathbf{Tr}(Q_{\mathrm{xx}} V_{\mathrm{xx}}(N)) \\
& + \textstyle\sum_{k=0}^{N-1} \mathbf{Tr}(Q_{\mathrm{xx}} V_{\mathrm{xx}}(k) + Q_{\mathrm{uu}} Z(k)) \\
\text{s.t.} \quad & \\
& V_{\mathrm{xx}}(0) \succcurlyeq I_n \\
& V_{\mathrm{xx}}(k) = X_{0,T} R(k) \\
& \begin{bmatrix} V_{\mathrm{xx}}(k+1) - I_n & X_{1,T} R(k) \\ R^\top(k) X_{1,T}^\top & V_{\mathrm{xx}}(k) \end{bmatrix} \succcurlyeq 0 \\
& \begin{bmatrix} Z(k) & Q_{\mathrm{uu}}^{1/2} U_{0,T} R(k) \\ R(k)^\top U_{0,T}^\top Q_{\mathrm{uu}}^{1/2} & V_{\mathrm{xx}}(k) \end{bmatrix} \succcurlyeq 0
\end{aligned} \tag{8}$$

*$\forall k \in \{0, \dots, N-1\}$, where*

$$\begin{aligned}
\mathcal{V}_{\mathrm{xx}} &:= \{V_{\mathrm{xx}}(1), \dots, V_{\mathrm{xx}}(N)\}, \\
\mathcal{R} &:= \{R(0), \dots, R(N-1)\}, \\
\mathcal{Z} &:= \{Z(0), \dots, Z(N-1)\}.
\end{aligned}$$

■

Similarly to section II-A, we can also rewrite (8) in SDP dual form. We emphasize that the computational cost of solving (8) grows with the size of the data.

## III. FIRST-ORDER ALGORITHMS FOR SDP IN DUAL FORM

In the spirit of the smoothing technique for non-smooth convex optimization [10], we introduce an accelerated first-order method for solving (7). In the first version of the algorithm, we fix the smoothing parameter ($\mu$) and change the stepsize ($\zeta$) adaptively and in a second version of the algorithm, we update both parameters adaptively.

### A. Smooth approximation

The problem in (7) is equivalent to

$$\begin{cases} \min_{y \in \mathbb{R}^\alpha} \; c^\top y \\ \qquad \text{s.t.} \quad \lambda_n(G_0 + \mathcal{G}^*(y)) \geq 0 \\ \qquad\qquad\; Py = b \end{cases} \tag{9}$$

where $\lambda_1 \geq \dots \geq \lambda_n$ are the eigenvalues of $G_0 + \mathcal{G}^*(y)$ in (9). Since the objective function in (9) is affine in $y$, the solution is attained on the boundary of the feasible region. Thus (9) can be written as

$$\begin{aligned}
&\min_{y \in Py=b} c^\top y \quad \text{s.t.} \quad \lambda_n(G_0 + \mathcal{G}^*(y)) = 0 \iff \\
&\min_{y \in Py=-b} -c^\top y \quad \text{s.t.} \quad \lambda_1(\mathcal{G}^*(y) - G_0) = 0.
\end{aligned} \tag{10}$$

The equality constraints in (10) are well suited for applying the method of multipliers. However, the function $\lambda_1(\cdot)$ is not differentiable; a common approach is then to use Nesterov's smooth approximation of $\lambda_1(\cdot)$ [10].

**Lemma 3 (Smoothness regularity)** [10] *Let $X \in \mathbb{S}^n$ and $\mu \in \mathbb{R}^+$. Then the function*

$$f_\mu(X) = \mu \log \left( \sum_{i=1}^{n} \exp(\lambda_i(X)/\mu) \right) \tag{11}$$

*is convex and twice differentiable with gradient*

$$\nabla f_\mu(X) = \left( \sum_{i=1}^{n} \exp(\lambda_i(X)/\mu) \right)^{-1} \sum_{i=1}^{n} \exp(\lambda_i(X)/\mu) q_i q_i^\top$$

*where $q_i$ is the $i^{\text{th}}$ column of the unitary matrix $Q$ in the eigen-decomposition $Q \Sigma Q^T$ of $X$. In addition, $f_\mu(X)$ satisfies the following inequalities due to the Lipschitz continuity of $\lambda_1$ :*

$$\lambda_1(X) \leq f_\mu(X) \leq \lambda_1(X) + \mu \log n. \tag{12}$$

*Therefore, $\lim_{\mu \to 0} f_\mu(X) = \lambda_1(X)$.*
*Also, the gradient $\nabla f_\mu(G_0 + \mathcal{G}^*(y))$ is Lipschitz continuous with constant $\|\mathcal{G}\|^2/\mu$, where $\|\mathcal{G}\| = \max_{h \in \mathbb{R}^m} \{ \|\mathcal{G}(y)\|_\infty \mid \|h\| = 1 \}$.* ■

By using a smooth approximation of $\lambda_1$, we can apply the method of multipliers to a smoothed variant of (10), i.e.

$$\min_{y \in Py=-b} \mathcal{L}_\mu(y, \nu; \sigma) \tag{13}$$

where

$$\mathcal{L}_\mu(y, \nu; \sigma) = -c^\top y - \nu f_\mu(\mathcal{G}^*(y) - G_0) + \frac{1}{2\sigma} f_\mu(\mathcal{G}^*(y) - G_0)^2$$

The following Algorithm 1 summarizes the augmented Lagrangian method for solving (13) [20, Chapter 17]. This algorithm has two loops where the inner loop minimizes the penalized function and the outer one updates the Lagrangian multiplier. In practice, it is extremely important that the inner loop algorithm is very fast and computationally inexpensive to speed up convergence. For this specific purpose, next we propose two novel accelerated first-order methods.

### B. Adaptive stepsize method

We first consider the application of an adaptive accelerated gradient descent method for minimizing a smooth convex function [21], where the stepsize is chosen adaptively. Namely, we fix $\mu$ (smoothing parameter) based on desired error in (12), and then minimize $\mathcal{L}_\mu(\cdot, \nu^t; \sigma_t)$ with

**Algorithm 1** Method of multipliers for equality constraints

```
Require: $\mu > 0$, $\sigma_0 > 0$, tolerance $\varepsilon > 0$, starting points $y_0^s$ and $\nu^0$
1: for $t = 0, 1, 2, \dots$ do
2:     compute $y_t = \min_{Py=-b} \mathcal{L}_\mu(y, \nu^t; \sigma_t)$ by starting at $y_t^s$
       (terminate if $\|\nabla_y \mathcal{L}_\mu(y, \nu^t; \sigma_t)\| \le \varepsilon$ or after specific iterations)   ▷Inner loop
3:     if final convergence test satisfied then
4:         STOP with approximate solution $y_k$
5:     end if
6:     Update Lagrange multipliers:
       $\nu^{t+1} = \nu^t - \frac{1}{\sigma_t} f_\mu(\mathcal{G}^*(y) - G_0)$
7:     Choose new penalty parameter $\sigma_{t+1} \in (0, \sigma_t)$
8:     Set starting point for the next iteration to $y_{t+1}^s = y_t$
9: end for
```

an adaptive stepsize. The following algorithm can be used to minimize $\mathcal{L}_\mu(\cdot, \nu^t; \sigma_t)$ in the inner loop of Algorithm 1:

$$\begin{cases} x_0 = x_1,\ \beta_0 = 0, \\ \beta_{k+1} = (1 + \sqrt{1 + 4\beta_k^2})/2,\ \gamma_k = (1 - \beta_k)/\beta_{k+1}, \\ \zeta_k : \max\limits_{\zeta \le \zeta_{k-1}} \Big\{\zeta \mid \nabla \mathcal{L}_\mu(x_k, \nu^t; \sigma_t)^\top \\ \qquad \nabla \mathcal{L}_\mu(x_k - \zeta \nabla \mathcal{L}_\mu(x_k, \nu^t; \sigma_t), \nu^t; \sigma_t) \\ \qquad -\frac{1}{2}\|\nabla \mathcal{L}_\mu(x_k, \nu^t; \sigma_t)\|^2 \ge 0\Big\} \quad \triangleright \textbf{LineSearch} \\ y_{k+1} = x_k - \zeta_k \nabla \mathcal{L}_\mu(x_k, \nu^t; \sigma_t), \\ x_{k+1} = (1 - \gamma_k) y_{k+1} + \gamma_k y_k \end{cases} \tag{Algorithm 2}$$

Unlike Nesterov's algorithm, the stepsize used in the above algorithm is not fixed. In Algorithm 2, the stepsize begins with a large value and it is non-increasing. Additionally, the linesearch in Algorithm 2 establishes the lower bound for the stepsize at $1/2L$, where $L$ represents a smooth constant of the objective function [21]. The adaptively changing stepsize contributes to the acceleration of the algorithm towards achieving the best theoretical convergence rate.

### *C. Adaptive-smoothing adaptive-stepsize method*

Now we introduce an extension of Algorithm 2 where we have an adaptive smoothing parameter besides the adaptive stepsize (the difference is just in the stepsize and the smoothing parameter update). The idea is that an adaptive smoothing of $\lambda_1(\cdot)$ in (10) should result in better convergence speed and approximation error.

**Theorem 1 (Convergence of adaptive smoothing)** *Consider the non-smooth function $f(\cdot) = \lambda_1(\cdot)$. Let $\varepsilon > 0$ be a desired precision and $T$ the number of iterations of Algorithm 3. If $T \ge D/\varepsilon$, then $f(y_T) - f^\star \le \varepsilon$ where*

$$D = 2\sqrt{2\log n\,(\|u_1\|^2 + \zeta_0\beta_0^2\delta_1 + (ab - b + a)\mu_0 \log n \zeta_0 \beta_0^2)},$$

*in which $\delta_1 = f_{\mu_1}(y_1) - f^\star$ and $u_1 = \beta_1 x_1 - (\beta_1 - 1)y_1 - x^\star$, that is, they only depend on the initial conditions and optimal solution.* ■

$$\begin{cases} \mu_{k+1} = \max\Big\{\dfrac{b\mu_k}{\dfrac{b(a-1)+a}{a-1}\dfrac{\beta_{k+1}^2}{\beta_k^2} - 1}, \dfrac{\varepsilon}{2\log n}\Big\} \ {\scriptstyle (a>1,\, b>0)} \\ \zeta_k : \max\limits_{\zeta \le \zeta_{k-1}} \{\zeta \mid \nabla f_{\mu_{k+1}}(x_k)^\top \nabla f_{\mu_{k+1}}(x_k - \zeta \nabla f_{\mu_{k+1}}(x_k)) \\ \qquad -\frac{1}{2}\|\nabla f_{\mu_{k+1}}(x_k)\|^2 - \frac{\mu_{k+1}}{\zeta}\log n \ge 0\} \triangleright \textbf{LineSearch} \\ y_{k+1} = x_k - \zeta_k \nabla f_{\mu_{k+1}}(x_k), \\ x_{k+1} = (1 - \gamma_k) y_{k+1} + \gamma_k y_k \end{cases} \tag{Algorithm 3}$$

To improve the convergence speed, Algorithm 3 can be used in the inner loop of Algorithm 1 to minimize $\mathcal{L}_\mu(\cdot, \nu^t; \sigma_t)$.

**Remark 1 (Convergence of adaptive stepsize)** *The same convergence rate can be proven for Algorithm 2 by fixing $\mu = \varepsilon/(2\log n)$.* ■

## IV. NUMERICAL SIMULATIONS

In this section, we compare our Algorithms 2 and 3 with Nesterov's accelerated gradient descent (NAGD) when used in the inner loop of Algorithm 1. The numerical performance of these algorithms are evaluated on a batch reactor system, a discretized version [22] with sampling time $0.1s$, which has unstable open-loop dynamics:

$$[\ A \mid B\ ] = \left[\begin{array}{cccc|cc} 1.178 & 0.001 & 0.511 & -0.403 & 0.004 & -0.087 \\ -0.051 & 0.661 & -0.011 & 0.061 & 0.467 & 0.001 \\ 0.076 & 0.335 & 0.560 & 0.382 & 0.213 & -0.235 \\ 0 & 0.335 & 0.089 & 0.849 & 0.213 & -0.016 \end{array}\right]$$

In the simulations, we take the cost weights $V_{\text{xx}} = I_4$ and $V_{\text{uu}} = I_2$. The inputs and initial conditions for the states, and the initial condition for the optimization algorithms are randomly generated with a normal distribution. The smoothing parameter ($\mu$) for NAGD and Algorithm 2 is set to 0.01, and the initial value of the Lagrangian coefficient ($\sigma$) is chosen randomly (the number of outer-loop iterations in Algorithm 1 is fixed to $t = 1, 2, \cdots, 10$). All random data are the same for the different algorithms and the various simulation scenarios.

To compare the algorithms, we consider two scenarios. In the first one, we terminate the first-order methods in the inner loop of Algorithm 1 after 300 iterations by fixing the number of data ($T = 15$) and the control horizon ($N = 10$). Figure 1 illustrates the histogram of objective cost in (8) by running the system with the achieved controllers with the solution of Algorithm 1 for 10 seconds. We see that Algorithm 3 reduces the objective function more than other algorithms. The Riccati optimal controller, constructed using the exact dynamics of the system, is used as a benchmark. In the second scenario, we change the length of the data and of the control horizon from $T = 10$ to $T = 30$ and from $N = 10$ to $N = 20$, respectively. In this case, the optimization method in the inner loop of Algorithm 1 is terminated if $\|\nabla_y \mathcal{L}_\mu(y, \nu^t; \sigma_t)\| \le \varepsilon = 10^{-5}$. Figure 2a shows the number of iterations versus the control horizon when the length of data is changed from $T = 10$ to $T = 30$. Remarkably,

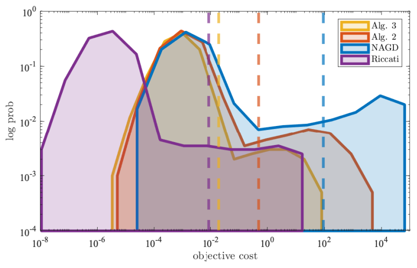
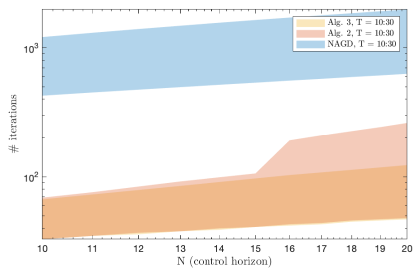
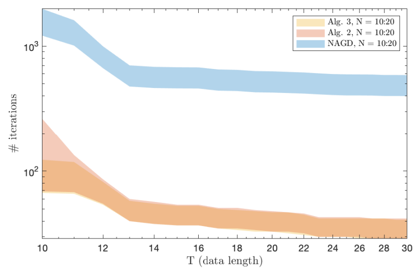


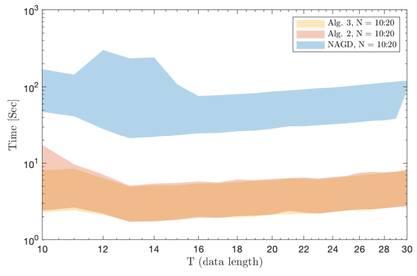
Fig. 1: Frequency plot of the objective cost.



(a)



(b)



(c)

Fig. 2: Simulation results for different cases.

Algorithm 2 and Algorithm 3 need fewer iterations to achieve the desired error. In another experiment, in Figure 2b, we plot the number of iterations needed to reach the desired error versus the size of the data by fixing the control horizon from $N = 10$ to $N = 20$. An interesting observation is that the more data we have, the fewer iterations are needed to reach the desired error. We note that by increasing the size of the data the complexity of each iteration increases, yet we observe overall no significant change in the simulation time (Figure 2c).

## V. Conclusion and Future Work

The data-driven finite-horizon linear quadratic regularization problem can be solved very efficiently and very accurately via semidefinite programming with adaptive accelerated first-order methods. Interestingly, by increasing the size of the experimental data, the number of iterations for solving the corresponding semidefinite program decreases. One problem associated with the smoothing parameter is its independency from the optimization variables. Breaking this independency via a suitable update rule is left as future work.

## Appendix

### A. *Proof of Theorem 1*

First, we show the function reduction of $f(\cdot)$ and its smooth approximation $f_\mu$.

$$\begin{aligned}
&f_{\mu_{k+1}}(y_{k+1}) - f(z) \overset{\text{(Lemma (3))}}{\leq} \\
&\qquad f_{\mu_{k+1}}(y_{k+1}) - f_{\mu_{k+1}}(z) + \mu_{k+1}\log n \\
&\leq f_{\mu_{k+1}}(y_{k+1}) - f_{\mu_{k+1}}(x_k) \\
&\qquad + \nabla f_{\mu_{k+1}}(x_k)^\top (x_k - z) + \mu_{k+1}\log n \\
&\leq \ \nabla f_{\mu_{k+1}}(y_{k+1})^\top (y_{k+1} - x_k) \\
&\qquad - \frac{1}{2}\nabla f_{\mu_{k+1}}(x_k)^\top (y_{k+1} - x_k) + \mu_{k+1}\log n \\
&+ \nabla f_{\mu_{k+1}}(x_k)^\top (x_k - z) + \frac{1}{2}\nabla f_{\mu_{k+1}}(x_k)^\top (y_{k+1} - x_k) \leq \\
&- \frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 - \frac{1}{\zeta_k}(y_{k+1} - x_k)^\top (x_k - z)
\end{aligned} \tag{14}$$

The last inequality of (14) holds due to the condition in the linesearch. By rewriting inequality (14) with $z = y_k$ and $z = x^*$, we have

$$\begin{aligned}
&f_{\mu_{k+1}}(y_{k+1}) - f_{\mu_k}(y_k) - \mu_k \log n \overset{\text{(Lemma (3))}}{\leq} \\
&f_{\mu_{k+1}}(y_{k+1}) - f(y_k) \leq -\frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 \\
&- \frac{1}{\zeta_k}(y_{k+1} - x_k)^\top (x_k - y_k)
\end{aligned} \tag{15}$$

$$\begin{aligned}
f_{\mu_{k+1}}(y_{k+1}) - f^* \leq -&\frac{1}{2\zeta_k}\|y_{k+1} - x_k\|^2 \\
&- \frac{1}{\zeta_k}(y_{k+1} - x_k)^\top (x_k - x^*)
\end{aligned} \tag{16}$$

Now by defining $\delta_k := f_{\mu_k}(y_k) - f^*$, multiplying (15) by $\beta_k - 1$, and adding the result to (16) we have

$$\begin{aligned}
\beta_k \delta_{k+1} - &(\beta_k - 1)\delta_k - \mu_k \log n(\beta_k - 1) \leq \\
&- \frac{\beta_k}{2\zeta_k}\|y_{k+1} - x_k\|^2 \\
&- \frac{1}{\zeta_k}(y_{k+1} - x_k)^\top (\beta_k x_k - (\beta_k - 1)y_k - x^*)
\end{aligned}$$

Multiplying above inequality by $\zeta_k\beta_k$, defining $\beta_{k-1}^2 := \beta_k^2 - \beta_k$ and using the fact that $\zeta_k \leq \zeta_{k-1}$ we can write

$$\begin{aligned}
&\zeta_k\beta_k^2\delta_{k+1} - \zeta_{k-1}\beta_{k-1}^2\delta_k - \mu_k \log n \zeta_k \beta_{k-1}^2 \leq \\
&\qquad\qquad - \frac{1}{2}(\|\beta_k(y_{k+1} - x_k)\|^2 \\
&+ 2\beta_k(y_{k+1} - x_k)^\top (\beta_k x_k - (\beta_k - 1)y_k - x^*))
\end{aligned} \tag{17}$$

Now one can verify that

$$\begin{aligned}&\|\beta_k(y_{k+1}-x_k)\|^2\\&+2\beta_k(y_{k+1}-x_k)^\top(\beta_k x_k-(\beta_k-1)y_k-x^*)=\\&\|\beta_k y_{k+1}-(\beta_k-1)y_k-x^*\|^2\\&\qquad\qquad-\|\beta_k x_k-(\beta_k-1)y_k-x^*\|^2\end{aligned}\tag{18}$$

Then by using (17) and (18), we obtain

$$\begin{aligned}&\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-\mu_k\log n\zeta_k\beta_{k-1}^2\le\\&\qquad-\frac{1}{2}(\|\beta_k y_{k+1}-(\beta_k-1)y_k-x^*\|^2\\&\qquad-\|\beta_k x_k-(\beta_k-1)y_k-x^*\|^2)\end{aligned}\tag{19}$$

In the right-hand terms of (19), we can drive the update rule of $x_{k+1}$,

$$\beta_k y_{k+1}-(\beta_k-1)y_k-x^*=\beta_{k+1}x_{k+1}-(\beta_{k+1}-1)y_{k+1}-x^*\tag{20}$$

which is equivalent to

$$x_{k+1}=\frac{(-1+\beta_k+\beta_{k+1})}{\beta_{k+1}}y_{k+1}+\frac{1-\beta_k}{\beta_{k+1}}y_k$$

By combining (19) and (20) with $u_k=\beta_k x_k-(\beta_k-1)y_k-x^*$ we obtain

$$\begin{aligned}&\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-\mu_k\log n\zeta_k\beta_{k-1}^2\le\\&\qquad\frac{1}{2}(\|u_k\|^2-\|u_{k+1}\|^2)\end{aligned}\tag{21}$$

We want to reduce $\mu_k$ in each iteration to reach our desired error. Therefore we know that $\mu_k<\mu_{k-1}$. Then we have

$$\begin{aligned}&-a\mu_{k-1}\log n\zeta_{k-1}\beta_{k-1}^2+(a-1)\mu_k\log n\zeta_k\beta_{k-1}^2\le\\&\quad-a\mu_k\log n\zeta_k\beta_{k-1}^2+(a-1)\mu_k\log n\zeta_k\beta_{k-1}^2\\&\quad=-\mu_k\log n\zeta_k\beta_{k-1}^2\end{aligned}\tag{22}$$

Let us define $\mu_k$ as follows

$$\mu_k=\left(\frac{b(a-1)+a}{a-1}\frac{\beta_k^2}{\beta_{k-1}^2}\mu_k-b\mu_{k-1}\right)$$

By substituting $\mu_k$ in the left-hand side of (22) and using (21) we have

$$\begin{aligned}&\zeta_k\beta_k^2\delta_{k+1}-\zeta_{k-1}\beta_{k-1}^2\delta_k-(ab-b+a)\mu_{k-1}\log n\zeta_{k-1}\beta_{k-1}^2\\&+(ab-b+a)\mu_k\log n\zeta_k\beta_k^2\le\frac{1}{2}(\|u_k\|^2-\|u_{k+1}\|^2)\end{aligned}\tag{23}$$

Summing inequalities (23) from $k=1$ to $k=T$ one obtains

$$\begin{aligned}&\zeta_T\beta_T^2\delta_{T+1}-\zeta_0\beta_0^2\delta_1-(ab-b+a)\mu_0\log n\zeta_0\beta_0^2\le\\&\qquad\frac{1}{2}\|u_1\|^2-\frac{1}{2}\|u_{T+1}\|^2\le\frac{1}{2}\|u_1\|^2\end{aligned}$$

which implies

$$\delta_{T+1}\le\frac{\|u_1\|^2+\zeta_0\beta_0^2\delta_1+(ab-b+a)\mu_0\log n\zeta_0\beta_0^2}{2\zeta_T\beta_T^2}\tag{24}$$

By the definition of $\delta_{T+1}$ and Lemma 3, we then have

$$\begin{aligned}&f(y_{T+1})-f^*\le\log n\,\mu_{T+1}+\\&\frac{\|u_1\|^2+\zeta_0\beta_0^2\delta_1+(ab-b+a)\mu_0\log n\zeta_0\beta_0^2}{2\zeta_T\beta_T^2}\end{aligned}\tag{25}$$

To achieve $\varepsilon$ error, we know that $\zeta_k$ has a lower bound $\varepsilon/(2\log n)$, and $\mu_k$ is decreasing until it reaches $\varepsilon/(2\log n)$. Then, the second term in the right-hand side of (25) is equal to $\varepsilon/2$, and thanks to the fact that $\beta_k\ge k/2$ for all $k$, we conclude that the $\varepsilon$ precision is guaranteed if $T$ is greater than

$$\frac{2\sqrt{2\log n\,(\|u_1\|^2+\zeta_0\beta_0^2\delta_1+(ab-b+a)\mu_0\log n\zeta_0\beta_0^2)}}{\varepsilon}.$$

■

## References

[1] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1 1994.

[2] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, pp. 442–458, 6 2019.

[3] L. Vandenberghe and S. Boyd, "Applications of semidefinite programming," *Applied Numerical Mathematics*, vol. 29, pp. 283–299, 3 1999.

[4] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," *SIAM Journal on Optimization*, vol. 5, pp. 13–51, 2 1995.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 3 2004.

[6] A. d'Aspremont and N. E. Karoui, "A stochastic smoothing algorithm for semidefinite programming," *SIAM Journal on Optimization*, vol. 24, pp. 1138–1177, 1 2014.

[7] A. Nemirovski and D. Yudin, *Problem complexity and method efficiency in optimization*. John Wiley, 1983.

[8] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, pp. 167–175, 5 2003.

[9] Y. Nesterov, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, pp. 127–152, 5 2005.

[10] ——, "Smoothing technique and its applications in semidefinite optimization," *Mathematical Programming*, vol. 110, pp. 245–259, 3 2007.

[11] ——, "A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$," *Proceedings of the USSR Academy of Sciences*, vol. 269, pp. 543–547, 1983.

[12] A. Yurtsever, O. Fercoq, F. Locatello, and V. Cevher, "A conditional gradient framework for composite convex minimization with applications to semidefinite programming," 2018, pp. 5713–5722.

[13] Q. Tran-Dinh, "Adaptive smoothing algorithms for nonsmooth composite convex minimization," *Computational Optimization and Applications*, vol. 66, pp. 425–451, 4 2017.

[14] M. Rotulo, C. D. Persis, and P. Tesi, "Data-driven linear quadratic regulation via semidefinite programming," *IFAC-PapersOnLine*, vol. 53, pp. 3995–4000, 2020.

[15] G. R. G. da Silva, A. S. Bazanella, C. Lorenzini, and L. Campestrini, "Data-driven lqr control design," *IEEE Control Systems Letters*, vol. 3, pp. 180–185, 1 2019.

[16] D. Bertsimas and B. V. Parys, "Sparse high-dimensional regression: Exact scalable algorithms and phase transitions," *The Annals of Statistics*, vol. 48, 2 2020.

[17] A. Gattami, "Generalized linear quadratic control," *IEEE Transactions on Automatic Control*, vol. 55, pp. 131–136, 1 2010.

[18] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 6 2013.

[19] C. D. Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Transactions on Automatic Control*, vol. 65, pp. 909–924, 3 2020.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer New York, 2006.

[21] R. R. Baghbadorani, S. Grammatico, and P. Mohajerin Esfahani, "Adaptive accelerated composite minimization," *preprint available at arXiv:2405.03414*, 2024.

[22] M. Green and D. J. N. Limebeer, *Linear robust control*. Prentice Hall, Englewood Cliffs, N.J., 1995.