Accelerating Model-Free Policy Optimization Using Model-Based Gradient: A Composite Optimization Perspective

Yansong Li$^*$ and Shuo Han$^1$

$^1$University of Illinois at Chicago

March 23, 2022

Abstract

We develop an algorithm that combines model-based and model-free methods for solving a nonlinear optimal control problem with a quadratic cost in which the system model is given by a linear state-space model with a small additive nonlinear perturbation. We decompose the cost into a sum of two functions, one having an explicit form obtained from the approximate linear model, the other being a black-box model representing the unknown modeling error. The decomposition allows us to formulate the problem as a composite optimization problem. To solve the optimization problem, our algorithm performs gradient descent using the gradient obtained from the approximate linear model until backtracking line search fails, upon which the model-based gradient is compared with the exact gradient obtained from a model-free algorithm. The difference between the model gradient and the exact gradient is then used for compensating future gradient-based updates. Our algorithm is shown to decrease the number of function evaluations compared with traditional model-free methods both in theory and in practice.

1 Introduction

Learning-based algorithms have flourished in the field of control [19] in recent years with applications in robotic manipulation and locomotion [12,13], energy systems [7], and transportation [23]. Learning-based control algorithms can be classified into two categories: model-free algorithms and model-based algorithms. Model-free algorithms do not utilize an explicit form of the dynamics and can handle dynamics that are hard to model. In the meantime, model-free algorithms require a large amount of data to learn an optimal policy. This may become impractical for physical systems, in which data collection is often expensive since every collection involves physical interactions with the environment.

In contrast, model-based algorithms maintain an explicit form of the dynamics such as a state-space model. With the aid of a model, model-based algorithms usually require less amount of data [22] to learn an optimal policy. However, model-based algorithms tend to have difficulties in modeling complex dynamics and may suffer from model bias [1,2,9] when the model class is not sufficiently rich.

In this paper, we develop a method that combines model-free and model-based methods to learn an optimal policy for controlling a nonlinear system under a quadratic cost. We formulate the optimal control problem as a composite optimization problem, in which the cost is expressed as a

$^*$Email: yli340@uic.edu
composite function defined by the sum of a model-based part and a model-free part: the model-based part has an analytical form, whereas the model-free part is viewed as a black box. We then develop a hybrid gradient-based algorithm that searches for the optimal solution using the model gradient, i.e., the gradient of the model-based part, plus a corrective compensation term, where the compensation is intermittently updated by the gradient of the model-free part. The algorithm is shown to require fewer function evaluations compared with purely model-free algorithms both in theory and in practice.

Related work  There have been several attempts in combining model-based and model-free algorithms. Qu et al. [18] considered a modification of the linear quadratic regulator (LQR) problem, where a small additive nonlinear perturbation is introduced to the original linear dynamics. They first applied model-based control using the approximate model given by the linear part of the dynamics to obtain a near-optimal policy. Then, they showed that, using the near-optimal policy as the initial policy, the model-free policy gradient method is able to produce an optimal policy for the modified LQR problem. Chebotar et al. [6] developed an algorithm to learn time-varying linear-Gaussian controllers for reinforcement learning. They proved that the update of a model-free algorithm, policy improvement with path integrals ($\text{PI}^2$), can be decomposed into two components, one depending on an approximate cost and another depending on the cost residual. The two component are updated separately, where the first component is updated with a model-based algorithm named LQR-FLM, and the second with $\text{PI}^2$. Shashua et al. [21] used a descent algorithm with a gradient mapping that is updated in each iteration based on the model and interactions with the environment. Our algorithm is closest to the one introduced by Abbeel et al. [1]. They used inaccurate dynamics to train a near-optimal policy, which was used to collect trajectories from the real Markov decision process (MDP). For each state-action pair from the collected trajectories, they compared the next state in the trajectories with the one predicted by the inaccurate MDP dynamics and used the difference to update the inaccurate MDP dynamics by adding a bias term. In comparison, our algorithm focuses on the control cost rather than the system dynamics. Specifically, our algorithm decomposes the control cost into two parts, one of which, called the model-based part, is related to inexact system dynamics. Instead of updating the inaccurate system dynamics, our algorithm updates the gradient of the model-based part of the cost using information from the true dynamics when performing gradient-based policy optimization.

Our work is also related to policy gradient for LQR, discussed in [5][10], in that we also use the same zeroth-order method to compute the gradient of the cost. See [8] for a comprehensive coverage of zeroth-order methods in optimization.

Finally, our work is closely related to composite optimization (see footnote 1). A classical method for solving composite optimization problems is the proximal gradient method [3][15][17][20]. The method assumes that the objective function can be expressed as the sum of a differentiable convex function and another possibly non-differentiable convex function. Moreover, the non-differentiable convex function is assumed to be associated with a proximal operator that can be efficiently evaluated. In contrast, our algorithm deals with composite optimization problems in which both functions in the sum are differentiable but does not assume the existence of an efficiently computable proximal operator.

1Throughout this paper, the term composite optimization refers to additive composite optimization, in which the objective function can be expressed as a sum of two functions [15]. The same term sometimes also refers to the case in which the objective function is a composition of two functions (see, e.g., [14]).
2 Background: Nonlinear optimal control with quadratic cost

We consider a modified discrete-time LQR problem, which appeared in [18]. The dynamical system is described by a nonlinear state-space model with state $x_t \in \mathbb{R}^n$ and control input $u_t \in \mathbb{R}^p$,

$$x_{t+1} = Ax_t + Bu_t + h(x_t, u_t),$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$ and $h: \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ with $h(0,0) = 0$. The nonlinear function $h$ is not assumed to admit an explicit form but is considered “small,” where the precise meaning of “small” will be discussed in Section 4.3. The goal is to find a linear state feedback controller $u_t = -Kx_t$ (see footnote 2) that minimizes the cost $C: \mathbb{R}^{p \times n} \to \mathbb{R}$ defined by

$$C(K) = \mathbb{E}_{x_0 \sim D} \left( \sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t \right),$$

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{p \times p}$ are positive definite and $x_0$ is the initial state and is drawn from a fixed distribution $D$. Multiple problems can be formulated as (1). For example, the dynamics of a robotic manipulator can be described by a linear state-space model plus an error term $h$. Note that if $h$ is zero everywhere, i.e., $h \equiv 0$, the problem is the same as the vanilla LQR problem, which has a closed-form solution.

Definition 1. We define $\hat{C}$ as the quadratic cost (2) when $h \equiv 0$ in (1). We denote the optimal point that minimizes $\hat{C}$ by $\hat{K}^*$ and the optimal point that minimizes $C$ by $K^*$.

Throughout this paper, we propose to minimize $C$ using a policy gradient method that uses gradient computed from a zeroth-order method [10], which only requires access to the values of the cost rather than an analytical expression of the gradient mapping. The method is model-free since it does not rely on an explicit form of the system dynamics. Because each evaluation of the cost function requires collecting trajectories generated from the dynamical system, model-free policy gradient methods need to sample numerous trajectories in order to find a near-optimal policy [22]. This issue of high sample complexity has been recognized as a major bottleneck in applying model-free policy gradient methods to physical systems, where the collection of trajectories are often expensive.

3 Proposed framework

To reduce the sample complexity of model-free policy gradient methods, we propose a policy optimization framework that leverages an inexact model of the system, given by the linear part of the dynamics. Our key idea is to reformulate the original policy optimization problem as a composite optimization problem that explicitly shows how model information enters policy optimization. The revelation of the role of the model naturally leads to an optimization algorithm that solves the composite optimization problem with fewer function evaluations than traditional model-free policy gradient methods.

---

2 The goal is not to find an optimal state feedback controller, which may be nonlinear for nonlinear systems even when the cost is quadratic.
3.1 A composite optimization perspective of policy optimization

We decompose the cost function $C$ into two components as

$$C(K) = \hat{C}(K) + r(K),$$

where $\hat{C}$ is defined in Definition 1 and $r$ is the residual term defined by $r \triangleq C - \hat{C}$. The problem of minimizing the cost in (3) is an unconstrained optimization problem of the following form:

$$\text{minimize}_{x \in \mathbb{R}^n} f(x),$$

where $f = \hat{f} + r$. We call $\nabla f(x)$ the exact gradient and $\nabla \hat{f}(x)$ the model gradient at $x$. Denote the optimal value of (4) by $f^*$ and an optimal solution of (4) by $x^*$. Also, denote $\hat{x}^*$ as a point where $\nabla \hat{f}(\hat{x}^*) = 0$.

We make the following assumptions throughout the paper.

**Assumption 1.** The function $f$ satisfies the PL-condition, i.e., there exists some $\mu > 0$ such that

$$\frac{1}{2} \|\nabla f(x)\|^2 \geq \mu(f(x) - f^*)$$

for all $x$.

**Assumption 2.** The residual mapping $r$ is $L_r$-smooth, i.e., $\nabla r$ is $L_r$-Lipschitz continuous.

**Assumption 3.** The gradient value $\nabla f(x)$ can be computed with $n$ function evaluations, where $n$ is the dimension of $x$. The model gradient $\nabla \hat{f}$ has a closed-form expression.

For the modified LQR problem defined by (1) and (2), Assumption 1 is satisfied when $K$ is close to $K^*$, since $C$ is locally strongly convex near $K^*$ [18], and strong convexity implies the PL-condition [11]. Assumption 2 is also satisfied when $K$ is close to $K^*$ due to the local smoothness of $C$ [18]. Assumption 3 is satisfied since $\nabla C(K)$ can be computed by zeroth-order methods, and $\nabla \hat{C}(K)$ has a closed-form solution [10].

3.2 A model-exploiting composite optimization algorithm

Throughout this paper, we focus on descent methods for solving the composite optimization problem (4). At each iteration, the value of the optimization variable is updated from $x$ to $x_+$ by

$$x_+ = x - t\Delta(x),$$

where $t$ is the step size, and $\Delta(x)$ is a descent direction satisfying $f(x - t\Delta(x)) < f(x)$ when $t$ is sufficiently small.

To solve the problem in (4), Qu et al. [18] used the gradient descent algorithm with the exact gradient starting from $x = \hat{x}^*$. The algorithm in [18] can be viewed as applying the gradient descent algorithm with the model gradient mapping $\nabla \hat{f}$ until $\nabla \hat{f}(x) = 0$, i.e., $x = \hat{x}^*$, after which the algorithm switches to using the exact gradient $\nabla f$. In our algorithm, shown in Algorithm 1, we also use $\nabla \hat{f}$, but we update $\nabla \hat{f}$ via a constant compensation $\delta$, which is defined as follows.

**Definition 2.** Consider a composite function $f = \hat{f} + r$. We call

$$\tilde{g} \triangleq \nabla \hat{f} + \delta$$

the compensated model gradient mapping compensated at point $\hat{x}$, where the compensation $\delta$ at $\hat{x}$ is a constant defined by

$$\delta \triangleq \nabla f(\hat{x}) - \nabla \hat{f}(\hat{x}) = \nabla r(\hat{x}).$$
Note that the model gradient mapping $\nabla \hat{f}$ can be viewed as a compensated model gradient mapping $\hat{g}$ with zero compensation. Due to the smoothness of $r$ (Assumption 2), when $\tilde{x}$ is close to $x$, $\nabla r(\tilde{x})$ will be close to $\nabla r(x)$. Thus, the compensated model gradient $\hat{g}(x) = \nabla \hat{f}(x) + \nabla r(\tilde{x})$ will be close to $\nabla f(x) = \nabla \hat{f}(x) + \nabla r(x)$. By Assumption 3, the constant compensation $\delta \triangleq \nabla f(\tilde{x}) - \nabla \hat{f}(\tilde{x}) = \nabla r(\tilde{x})$ can be computed with $n$ function evaluations.

Algorithm 1 Gradient compensation algorithm

1: Obtain $\eta$ by applying backtracking line search with the compensated model gradient mapping $\hat{g}$ at $x$, i.e., $\hat{g}(x)$.
2: if backtracking line search succeeds at $x$ then \(\triangleright\) Model-based Regime
3: \(x \leftarrow x - \eta \hat{g}(x)\)
4: else \(\triangleright\) Model-free Regime
5: Apply backtracking line search with the exact gradient $\nabla f(x)$ to obtain $\eta$.
6: $x \leftarrow x - \eta \nabla f(x)$
7: $\tilde{g} \leftarrow \nabla \hat{f} + \nabla f(x) - \nabla \hat{f}(x)$
8: end if

The difference in the use of model information between our algorithm and the algorithm in [18] is highlighted below (where “LSF” stands for “backtracking line search fails”):

- Qu et al. [18]: $\nabla \hat{f} \xrightarrow{\nabla f = 0} \nabla f$

- Our algorithm: $\nabla \hat{f} \xrightarrow{\nabla f = 0} \nabla f \xrightarrow{\hat{g}^{(1)} \text{ LSF}} \nabla f \xrightarrow{\hat{g}^{(2)} \text{ LSF}} \ldots$

The mapping $\hat{g}^{(i)}$ is the $i$-th compensated model gradient mapping. Our algorithm performs gradient descent using $\hat{g}^{(i)}$ until $\hat{g}^{(i)}(x)$ is no longer a descent direction at the current iterate $x$, which can be discovered from the failure of backtracking line search [4], i.e.,

$$f(x - \eta \hat{g}(x)) \geq f(x) - \alpha \eta \| \hat{g}(x) \|^2$$

for any step size $\eta > 0$ and $\alpha \in (0, 1/2)$. When backtracking line search fails, our algorithm performs one step of gradient descent using the exact gradient $\nabla f(x)$, replaces $\hat{g}^{(i)}$ with $\hat{g}^{(i+1)} \triangleq \nabla \hat{f} + \nabla f(x) - \nabla \hat{f}(x)$, and continues with the new compensated model gradient mapping $\hat{g}^{(i+1)}$. Note that the standard backtracking line search [4] will not terminate when it fails to find a positive step size $\eta$. We will revisit this issue in Section 4.1.

In the following, we say that the algorithm operates in the model-free regime when the exact gradient is used and in the model-based regime when the compensated model gradient mapping $\hat{g}$ is used. Let the superscript $i$ denote the index for the model-free iterations and the subscript $j$ the index for the model-based iterations between two model-free iterations, i.e.,

$$x_{j+1}^{(i)} = x_j^{(i)} - \eta \hat{g}^{(i)}(x_j^{(i)}) \quad \text{(model-based update)}$$

and

$$x_i^{(i+1)} = x_{N_i}^{(i)} - \eta \nabla f(x_{N_i}^{(i)}) \quad \text{(model-free update)},$$

where $N_i$ is the number of model-based iterations between the $(i-1)$-th and the $i$-th model-free iterations. We will drop the iteration index $i$ when the results do not depend on a specific model-free iteration.
4 Main result

When the exact gradient $\nabla f$ is used for computing the update $x_+$, our algorithm is identical to the vanilla gradient descent. Therefore, we only need to consider the case when $x_+$ is computed using the compensated model gradient mapping $\tilde{g}$ for analyzing the performance of our algorithm. In other words, we only need to consider the progress that our algorithm made in the model-based regime.

4.1 Modified backtracking line search

In the model-based regime, the compensated model gradient mapping $\tilde{g}$ is used in place of the exact gradient mapping $\nabla f$. As a result, the standard backtracking line search [4] may no longer terminate, as illustrated in the following example.

Example 1. Consider $f(x) = (x - 2)^2$ with a compensated model gradient $\tilde{g}(x) = 2x$. The compensated model gradient mapping $\tilde{g}$ satisfies $\tilde{g}(x) > 0$ when $x \in (0, 2)$. Also, $f$ is monotonically decreasing when $x < 2$, i.e., $f(x + \Delta x) \geq f(x)$ for any $\Delta x < 0$ and $x < 2$. So $f(x - \eta \tilde{g}(x)) \geq f(x) - \alpha \eta \|\tilde{g}(x)\|^2$ for any $\alpha > 0$ and $\eta > 0$. In other words, the Armijo condition [16, page 33], which is used for terminating the standard backtracking line search, cannot be satisfied for any positive $\eta$, implying that the standard backtracking line search [4] never terminates.

To ensure that the backtracking line search subroutine eventually terminates, we set a lower bound $\eta_{\text{min}}$ for the step size $\eta$. The algorithm quits the backtracking line search subroutine and reports that line search fails whenever $\eta \leq \eta_{\text{min}}$. The modified backtracking line search algorithm is presented as Algorithm 2.

We denote the initial step size in the backtracking line search as $\eta_{\text{max}}$. Upon successful termination, the modified backtracking line search always returns a bounded step size $\eta \in (\eta_{\text{min}}, \eta_{\text{max}})$. The modified backtracking line search will only be used in the model-based regime, whereas the standard backtracking line search will be used in the model-free regime. In the following sections, we will simply use the term backtracking line search when the type of backtracking line search is clear from the context.

Algorithm 2 BtLineSearch

Input: $x, f, \Delta, \alpha, \beta, \eta_{\text{min}}, \eta_{\text{max}}$

Output: $\eta$

1: $\eta \leftarrow \eta_{\text{max}}$
2: while $f(x - \eta \Delta(x)) > f(x) - \alpha \eta \|\Delta(x)\|^2$ do
3: if $\eta \geq \eta_{\text{min}}$ then
4: $\eta \leftarrow \beta \eta$
5: else
6: $\eta \leftarrow 0$ \hspace{1cm} \triangleright Line search fails.
7: end if
8: end while

Another issue we need to deal with, which will be shown in the next example, is that the algorithm may get stuck in the model-based regime and never converge to the optimal solution.

Example 2. Consider the case in Example 1. The update rule for this case is given by $x_{j+1} = x_j - \eta_j \tilde{g}(x_j) = (1 - 2\eta_j)x_j$ for $j = 1, \ldots, N$. When the update starts at $x_1 < 0$ with $\eta_{\text{max}} < 1/2$,
we have $x_j \leq (1 - 2\eta_{\text{max}})^j x_1 < 0$ for all $j$. Fix an $\alpha \in (0, 1/2)$, and suppose $\eta_{\text{min}} < 1 - \alpha - 2/x_1$. It can be verified that backtracking line search always succeeds since $f(x_{j+1}) < f(x_j) - \alpha \eta_j \|\tilde{g}(x_j)\|$ for $\eta_j \in (\eta_{\text{min}}, 1/2)$. Despite the success of line search, the algorithm never converges to the optimal solution $x^* = 2$.

Example 2 shows that an inappropriate choice of $\eta_{\text{max}}$ may cause the algorithm to converge to a suboptimal solution. In general, however, it is difficult to verify whether the choice of $\eta_{\text{max}}$ is appropriate. We will discuss in Section 4.2 how to deal with this non-convergent behavior.

### 4.2 The sufficient decrease condition

Example 2 shows that the algorithm may not “make sufficient progress” even when line search succeeds. In the following, we formally define what “sufficient progress” means and give a verifiable condition to test whether the algorithm makes sufficient progress when line search succeeds.

**Definition 3.** A point $x_+$ is said to satisfy the **sufficient decrease condition** for $f$ at $x$ if

$$f(x_+) < f(x) - t\|\nabla f(x)\|^2$$

for some $t > 0$.

In vanilla gradient descent, when backtracking line search succeeds, the Armijo condition

$$f(x - \eta \nabla f(x)) < f(x) - \alpha \eta \|\nabla f(x)\|^2 \quad (5)$$

is satisfied for some step size $\eta > 0$ and $\alpha \in (0, 1/2)$. It can be seen from (5) that $x_+ = x - \eta \nabla f(x)$ satisfies the sufficient decrease condition for $f$ at $x$. Because backtracking line search always succeeds when $\nabla f(x) \neq 0$, the condition in (5) always holds, implying that the function value will decrease sufficiently in every iteration. The sufficient decrease condition is central to proving the convergence of vanilla gradient descent [4] as well as our algorithm, which will be shown in Section 4.3. The following example shows that backtracking line search may not guarantee sufficient decrease when the model gradient is used.

**Example 3.** Consider again the case in Example 1. Set $x = -2$, $\alpha = 1/2$, and $\eta = 1/2$. Backtracking line search succeeds since $f(x - \eta \tilde{g}(x)) = 4 < 12 = f(x) - \alpha \eta \|\tilde{g}(x)\|^2$. Nevertheless, the sufficient decrease condition is not satisfied since $f(x - \eta \tilde{g}(x)) \geq 0 = f(x) - \alpha \eta \|\nabla f(x)\|^2$.

Although the success of backtracking line search does not in general imply the sufficient decrease condition, the following proposition shows that the implication holds when $\|\nabla f(x) - \tilde{g}(x)\|$ is sufficiently small.

**Proposition 1.** Suppose backtracking line search succeeds at $x$ when using the compensated model gradient mapping $\tilde{g}$, and $\tilde{g}$ satisfies

$$\|\nabla f(x) - \tilde{g}(x)\| \leq \frac{1 - \gamma}{\gamma} \|\tilde{g}(x)\| \quad \text{(6)}$$

for some $\gamma \in (0, 1)$. Then the updated point $x_+ = x - \eta \tilde{g}(x)$, where $\eta$ is given by backtracking line search, satisfies the sufficient decrease condition for $f$ at $x$. 


Proof. From the triangle inequality and (6), one can see that
\[ \|\nabla f(x)\| - \|\tilde{g}(x)\| \leq \|\nabla f(x) - \tilde{g}(x)\| \leq \frac{1-\gamma}{\gamma}\|\tilde{g}(x)\|. \]

Since \( \gamma > 0 \), it holds that
\[ \gamma(\|\nabla f(x)\| - \|\tilde{g}(x)\|) \leq (1 - \gamma)\|\tilde{g}(x)\|. \]

Thus,
\[ \|\tilde{g}(x)\| \geq \gamma\|\nabla f(x)\|. \]  \hspace{1cm} (7)

Recall that, when the backtracking line search algorithm returns a nonzero step \( \eta \neq 0 \), it holds that
\[ f(x_{+}) \leq f(x) - \alpha\eta\|\tilde{g}(x)\|^2. \]  \hspace{1cm} (8)

Combining (7) and (8), we have
\[ f(x_{+}) \leq f(x) - \alpha\gamma^2\eta\|\nabla f(x)\|^2, \]  \hspace{1cm} (9)

from which the proposition follows. \[ \square \]

Define \( e(x) \triangleq \|\nabla f(x) - \tilde{g}(x)\| \), the error of the compensated model gradient mapping \( \tilde{g} \) at \( x \). According to Proposition 1, backtracking line search will ensure a sufficient decrease for updates using the compensated model gradient mapping \( \tilde{g} \) when \( e(x) \) is small. Conceivably, one can determine whether the sufficient decrease condition is still met through monitoring the value of \( e(x) \). However, it is desirable to avoid evaluating \( e(x) \) directly since computing \( \nabla f \) is expensive.

Instead of evaluating \( e(x) \) directly, we propose to construct an upper bounding sequence \( \{\bar{e}_{j}\} \) satisfying \( \bar{e}_{j} \geq e(x_{j}) \) for all \( j = 1, \ldots, N \) and monitor \( \bar{e}_{j} \) in place of \( e(x_{j}) \). Specifically, we construct \( \{\bar{e}_{j}\} \) by choosing
\[ \bar{e}_{1} = 0, \quad \bar{e}_{j+1} = L_r\|x_{j+1} - x_j\| + \bar{e}_{j}. \]  \hspace{1cm} (10)

The following theorem shows that the sequence \( \{\bar{e}_{j}\} \) constructed in (10) can be used to guarantee the sufficient decrease condition when backtracking line search succeeds.

**Theorem 1.** Consider a sequence \( \{x_j\}_{j=1}^N \) defined by \( x_{j+1} = x_j - \eta_j\tilde{g}(x_j) \), where \( \eta_j \) is given by the backtracking line search. Suppose the backtracking line search using \( \tilde{g}(x_j) \) succeeds at \( x_j \) for all \( j \). Let \( \{\bar{e}_{j}\} \) be defined by (10). If for a fixed \( \gamma \in (0,1) \), we have
\[ \bar{e}_{j} \leq \frac{1-\gamma}{\gamma}\|\tilde{g}(x_{j})\| \]  \hspace{1cm} (11)

for any \( j \), then \( x_{j+1} \) satisfies the sufficient decrease condition for \( f \) at \( x_j \) for all \( j = 1, \ldots, N \).

Proof. By the triangle inequality and the smoothness of \( r \), we have
\[ e(x_{j+1}) = \|\nabla r(x_{j+1}) - \delta\| \]
\[ \leq \|\nabla r(x_{j+1}) - \nabla r(x_j)\| + \|\nabla r(x_j) - \delta\| \]
\[ \leq L_r\|x_{j+1} - x_j\| + e(x_j). \]

By induction and (10) we have \( \bar{e}(x_j) \geq e(x_j) \). The theorem follows from Proposition 1. \[ \square \]
We can augment Algorithm 1 by incorporating condition (11) and use the compensated model gradient mapping \( \tilde{g} \) to update \( x \) only when (11) is satisfied and backtracking line search succeeds in finding a positive step size. The factor \( \gamma \) can be viewed as a hyperparameter of the algorithm; a smaller \( \gamma \) will make (11) easier to satisfy thus making our algorithm stay for more iterations in the model-based regime. The augmented algorithm is presented as Algorithm 3. As long as \( \gamma \) is small enough, condition (11) will always be satisfied when the algorithm stays in the model-based regime at \( x_j \) unless \( \tilde{g}(x_j) = 0 \). When \( \tilde{g}(x_j) = 0 \), it implies that \( x_j \) is a stationary point as predicted by \( \tilde{g} \), at which point it becomes necessary to quit the model-based regime to verify whether \( x_j \) is truly a stationary point that satisfies \( \nabla f(x_j) = 0 \).

Algorithm 3 Gradient Compensation Algorithm

1: Start from \( x^{(1)} = \hat{x}^*, \bar{e}_0 = 0 \) and \( \delta_1 = \nabla r(\hat{x}^*) \).
2: for \( i = 1 : \text{iters} \) do
3: \( x^{(i+1)} \leftarrow \text{ModelBasedDescent}(x^{(i)}, \delta, \gamma, \nabla \hat{f}, \bar{e}_0) \)
4: \( \delta_{i+1} \leftarrow \nabla r(x^{(i+1)}) = \nabla f(x^{(i+1)}) - \nabla \hat{f}(x^{(i+1)}) \)
5: \( \eta \leftarrow \text{BtLineSearch}(x, f, \nabla f(x^{(i+1)}), \alpha, \beta, 0, \eta_{\text{max}}) \) \( \triangleright \) Standard backtracking line search
6: \( x_+^{(i+1)} \leftarrow x^{(i+1)} - \eta \nabla f(x^{(i+1)}) \)
7: \( \bar{e}_0 \leftarrow L_r \| x_+^{(i+1)} - x^{(i+1)} \| \)
8: \( x^{(i+1)} \leftarrow x_+^{(i+1)} \)
9: end for

Algorithm ModelBasedDescent

Input: \( x, \delta, \gamma, \nabla \hat{f}, \eta_{\text{min}}, \alpha, \beta, \bar{e}_0 \)

Output: \( x \)

1: \( x_+ \leftarrow x \)
2: \( \tilde{g} \leftarrow \nabla \hat{f} + \delta \)
3: \( \bar{e} \leftarrow \bar{e}_0 \)
4: while \( \bar{e} \leq \frac{1 - \gamma}{\gamma} \| \tilde{g} \| \) and \( \text{BtLineSearch}(x, f, \tilde{g}, \alpha, \beta, \eta_{\text{min}}, \eta_{\text{max}}) \neq 0 \) do
5: \( x \leftarrow x_+ \)
6: \( \tilde{g} \leftarrow \nabla \hat{f} + \delta \)
7: \( \eta \leftarrow \text{BtLineSearch}(x, f, \tilde{g}, \alpha, \beta, \eta_{\text{min}}, \eta_{\text{max}}) \) \( \triangleright \) Modified backtracking line search
8: \( x_+ \leftarrow x - \eta \tilde{g} \)
9: \( \bar{e} \leftarrow L_r \| x_+ - x \| + \bar{e} \)
10: end while

4.3 Progress in the model-based regime

Theorem 1 shows that under condition (11), the success of the backtracking line search implies a sufficient decrease in the objective function. An immediate consequence of sufficient decrease is that the suboptimality gap in the model-based regime decreases geometrically, as described in the following proposition.

Proposition 2. Suppose \( f \) satisfies Assumption 1, condition (11) holds, and the backtracking line search using the compensated model gradient mapping \( \tilde{g} \) succeeds at \( x_j \) for all \( j = 1, 2, \ldots N \). Then

\[
    f(x_{N+1}) - f^* \leq (1 - 2\mu \alpha \gamma^2 \eta_{\text{min}})^N (f(x_1) - f^*),
\]
where $\eta_{\text{min}}$ is the lower bound on the step size, and $N$ is the number of iterations in the model-based regime between two model-free iterations.

**Proof.** The proof is adopted from the one for [11, Theorem 1]. By Assumption 1 and (9),

$$f(x_{j+1}) - f^* \leq (1 - 2\mu\alpha\gamma^2\eta_j)(f(x_j) - f^*).$$  \hspace{1cm} (12)

Applying (12) recursively yields

$$f(x_{N+1}) - f^* \leq \prod_j (1 - 2\mu\alpha\gamma^2\eta_j)(f(x_1) - f^*) \leq (1 - 2\mu\alpha\gamma^2\eta_{\text{min}})^N(f(x_1) - f^*).$$

The relative error reduction in the model-based regime of our algorithm, according to Proposition 2, is given by $(1 - 2\mu\alpha\gamma^2\eta_{\text{min}})^N$, where $N$ is the total number of iterations in the model-based regime. Although it seems that a larger $N$ will lead to a larger error reduction, this is not true in general. To make the algorithm spend more iterations in the model-based regime, it is necessary to decrease either $\eta_{\text{min}}$ to prevent the failure of line search or $\gamma$ to make condition (11) easier to satisfy, both of which can potentially make $(1 - 2\mu\alpha\gamma^2\eta_{\text{min}})^N$ larger even if $N$ grows.

In Theorem 2 that will soon follow, we will show that the algorithm can ensure sufficient decrease for at least a few model-based iterations when backtracking line search succeeds. Before presenting the theorem, we need to introduce the following definition.

**Definition 4 (Bounded update direction).** An update direction mapping $\Delta$ is called a **bounded update direction** if there exist $\kappa_{\text{min}}$ and $\kappa_{\text{max}}$ satisfying $\kappa_{\text{min}} \in (0, 1)$ and $\kappa_{\text{max}} > \kappa_{\text{min}}$ such that for any $x$ and any bounded step size $\eta \in (\eta_{\text{min}}, \eta_{\text{max}})$, it holds that

$$\kappa_{\text{min}}\|\Delta(x)\| \leq \|\Delta(x_+)\| \leq \kappa_{\text{max}}\|\Delta(x)\|,$$

where $x_+ = x - \eta\Delta(x)$.

We give some examples in the following to show how the bounded update direction is related to our compensated model gradient mapping $\tilde{g}$.

**Example 4.** Consider the quadratic function $f$ defined by $f(x) = (1/2)(x^2 + c_2(x - c_3)^2)$. It can be shown that the compensated model gradient mapping $\tilde{g}$, given by $\tilde{g}(x) = x + \delta$, is a bounded updated direction. This is because

$$\tilde{g}(x_+) = x_+ + \delta = x - \eta\tilde{g}(x) + \delta$$

$$= x - \eta(x + \delta) + \delta = (1 - \eta)(x + \delta) = (1 - \eta)\tilde{g}(x).$$

Since $\eta \in (\eta_{\text{min}}, \eta_{\text{max}})$, it suffices to choose $\kappa_{\text{max}} = 1 - \eta_{\text{min}}$ and $\kappa_{\text{min}} = 1 - \eta_{\text{max}}$ to establish that $\tilde{g}$ is a bounded updated direction.

**Example 5.** Consider the function $f$ defined as

$$f(x) = \frac{c_1}{2}x^\top P x + \frac{c_2}{2}(x - c_3)^\top Q(x - c_3)$$

where $P$ and $Q$ are positive definite. Without loss of generality, we choose $c_1 = 1$. It can be shown that the compensated model gradient mapping $\tilde{g}$, given by $\tilde{g}(x) = Px + \delta$, is a bounded update direction: Let $x_+ = x - \eta\tilde{g}(x)$. It holds that $\tilde{g}(x_+) = (1 - \eta P)(Px + \delta) = (1 - \eta P)\tilde{g}(x)$. When $\eta \in (\eta_{\text{min}}, \eta_{\text{max}})$, one can choose $\kappa_{\text{max}} = 1 - \eta_{\text{min}}\lambda_{\text{min}}(P)$ and $\kappa_{\text{min}} = 1 - \eta_{\text{max}}\lambda_{\text{max}}(P)$, where $\lambda_{\text{max}}(P)$ and $\lambda_{\text{min}}(P)$ are the maximum and the minimum eigenvalues of $P$, respectively.
Intuitively, Algorithm 3 will stay in the model-based regime when the compensated model gradient \( \tilde{g}(x) \) is close to the exact gradient \( \nabla f(x) \). To lower bound the number of iterations the algorithm stays in the model-based regime, we need to bound the change of the compensated model gradient mapping \( \tilde{g} \) at each iterate, which is captured by the notion of bounded update direction introduced in Definition 4. Using the notion of bounded update direction, we present in the following theorem a lower bound on the number of model-based iterations.

**Theorem 2.** Let \( x_1 \) be the initial point when Algorithm 3 enters the model-based regime with a compensated model gradient mapping \( \tilde{g} \). Suppose \( \tilde{g} \) is a bounded updated direction mapping with constants \( \kappa_{\text{max}} \) and \( \kappa_{\text{min}} \), and \( \tilde{g}(x_1) \neq 0 \). Then the success of backtracking line search implies the sufficient decrease condition if \( N \) satisfies

\[
\log |\kappa_{\text{max}}^N - 1| + N \log \frac{1}{\kappa_{\text{min}}} \leq \log \frac{1 - \gamma}{\gamma \eta_{\text{max}} L_r} + \log |\kappa_{\text{max}} - 1|
\]

when \( \kappa_{\text{max}} \neq 1 \) and

\[
\log N + N \log \frac{1}{\kappa_{\text{min}}} \leq \log \frac{1 - \gamma}{\gamma \eta_{\text{max}} L_r}
\]

when \( \kappa_{\text{max}} = 1 \).

**Proof.** Suppose backtracking line search does not ensure sufficient decrease after \( N \) steps — that is,

\[
\bar{e}_{N+1} > \frac{1 - \gamma}{\gamma} \|\tilde{g}(x_{N+1})\|.
\]

When \( \kappa_{\text{max}} \neq 1 \),

\[
\bar{e}_{N+1} = L_r \sum_{j=1}^{N} \|x_{j+1} - x_j\| = L_r \sum_{j=1}^{N} \eta_j \|\tilde{g}(x_j)\| \leq L_r \eta_{\text{max}} \frac{1 - \kappa_{\text{max}}^N}{1 - \kappa_{\text{max}}} \|\tilde{g}(x_1)\|.
\]

Combine with equation (15) to obtain

\[
L_r \eta_{\text{max}} \frac{1 - \kappa_{\text{max}}^N}{1 - \kappa_{\text{max}}} \|\tilde{g}(x_1)\| > \frac{1 - \gamma}{\gamma} \|\tilde{g}(x_{N+1})\| \geq \frac{1 - \gamma}{\gamma} \kappa_{\text{min}} \|\tilde{g}(x_1)\|.
\]

Rearranging the above formula yields

\[
\log |\kappa_{\text{max}}^N - 1| + N \log \frac{1}{\kappa_{\text{min}}} > \log \frac{1 - \gamma}{\gamma \eta_{\text{max}} L_r} + \log |\kappa_{\text{max}} - 1|.
\]

Similarly, when \( \kappa_{\text{max}} = 1 \), by the triangle inequality,

\[
\bar{e}_{N+1} = L_r \sum_{j=1}^{N} \|x_{j+1} - x_j\| = L_r \sum_{j=1}^{N} \eta_j \|\tilde{g}(x_j)\| \leq L_r \eta_{\text{max}} N \|\tilde{g}(x_1)\|.
\]

Thus, the same procedure as above gives us

\[
\log N + N \log \frac{1}{\kappa_{\text{min}}} > \log \frac{1 - \gamma}{\gamma \eta_{\text{max}} L_r}.
\]

The theorem follows by taking the contrapositive of the results in (16) and (17).
The left-hand side of (13) and that of (14) are monotonically increasing with respect to \( N \). Let \( N_{\min} \) be the largest \( N \) satisfying (13) or (14), depending on the value of \( \kappa_{\max} \). Theorem 2 ensures that our algorithm makes sufficient progress in the model-based regime for at least \( N_{\min} \) iterations as long as backtracking line search succeeds. Because the right-hand side of (13) and (14) are monotonically decreasing as \( L_r \) grows, a smaller \( L_r \) is desired for making the algorithm stay for more iterations in the model-based regime when backtracking line search succeeds. Recall in Section 2 we require \( h \) to be “small” without formally defining how to quantify the magnitude of \( h \). Theorem 2 indicates that the smoothness constant \( L_r \) of the residual cost \( r \) is a useful metric for quantifying the magnitude of \( h \). Indeed, when the dynamics in (1) are nearly linear, the constant \( L_r \) will be small, and the algorithm is guaranteed to make better use of model information by staying in the model-based regime for more iterations. In the extreme case of \( h \equiv 0 \), the residual cost \( r \equiv 0 \) and hence \( L_r = 0 \), implying that the algorithm always stays in the model-based regime.

### 4.4 Number of function evaluations

In Section 4.3, we showed that Algorithm 3 will stay in the model-based regime for at least some iterations and make progress when backtracking line search succeeds. We will show in the following that the model-based regime requires fewer function evaluations than the model-free regime for making the same progress when the dimension \( n \) of the problem is large.

Recall that, when gradient descent is applied to strongly convex and smooth objectives, the objective values for two successive iterations \( i \) and \( i + 1 \) satisfy

\[
f(x_{i+1}) - f^* = \rho (f(x_i) - f^*),
\]

where \( \rho \) is the convergence ratio. After \( N \) iterations, the relative error reduction is given by \( \phi \triangleq (f(x_N) - f^*)/(f(x_0) - f^*) = \rho^N \). We propose to characterize the progress made per iteration by \( |\log \phi|/N \), which recovers \( |\log \rho| \) and hence is independent of \( N \) when (18) holds. Similarly, we define the progress made per function evaluation as \( |\log \phi|/(mN) \), where \( m \) is the number of function evaluations in each iteration.

By Assumption 3 at any point \( x \), the number of function evaluations for computing \( \nabla f(x) \) is \( n \), whereas the number of function evaluations for computing \( \nabla f(x) \) is zero. From Algorithm 2, the maximum number \( m_{\max} \) of function evaluations inside the backtracking line search subroutine is the smallest integer \( m \) satisfying \( \beta^{m-1} \eta_{\max} \leq \eta_{\min} \). Thus,

\[
m_{\max} = \left\lceil \frac{\log(\eta_{\min}/\eta_{\max})}{\log \beta} + 1 \right\rceil,
\]

which only depends on the parameters of backtracking line search. Also, from Proposition 2 the relative error reduction in the model-based regime is \((1 - 2\mu \alpha \gamma^2 \eta_{\min})^N \). Thus, the progress made per function evaluation in the model-based regime is lower bounded by

\[
\frac{|\log(1 - 2\mu \alpha \gamma^2 \eta_{\min})^N|}{m_{\max}N} = \frac{|\log(1 - 2\mu \alpha \gamma^2 \eta_{\min})|}{m_{\max}},
\]

which is a constant when the parameters of backtracking line search and \( \gamma \) are fixed. In the meantime, when \( f \) is assumed to be \( L_f \)-smooth [11], the convergence ratio \( \rho \) in the model-free regime is \( 1 - \mu/L_f \). Also, from Assumption 3 the number of function evaluations in each iteration in the model-free regime is at least \( n \), which includes \( n \) evaluations needed for computing the exact gradient and any additional evaluations used in backtracking line search. Thus, the progress made per function evaluation is upper bounded by

\[
\frac{|\log \rho|}{n} = \frac{|\log(1 - \mu/L_f)|}{n},
\]

(20)
which decreases with respect to the problem dimension $n$.

When the dimension $n$ of the problem is large enough, the quantity in (19) will be larger than that in (20), implying that more progress is made per function evaluation in the model-based regime. Since Algorithm 3 always makes progress in the model-based regime as discussed in Section 4.2, the algorithm is guaranteed to use fewer function evaluations than a purely model-free method for achieving the same level of suboptimality.

5 Numerical experiments

In this section, we shall refer to Algorithm 3 as GC, which stands for “gradient compensation.” We compare GC with the method in [18] in two different settings: (1) Both $\hat{f}$ and $r$ are convex quadratic functions. ($f = \hat{f} + r$ is convex and smooth.) (2) The objective function is given by $C$ in (3), and $\hat{f}$ is given by $\hat{C}$ from Section 2. ($C$ is non-convex and locally smooth.)

5.1 Quadratic functions

We consider the quadratic functions $\hat{f}$, $r$: $\mathbb{R}^n \to \mathbb{R}$ defined as $\hat{f}(x) = c_1 x^\top P x / 2$ and $r(x) = c_2 (x - c_3)^\top Q (x - c_3) / 2$, where both $P$ and $Q$ are positive definite. To ensure that $P$ is positive definite, we generated its square root $P^{1/2}$ randomly, with each entry drawn independently from the standard Gaussian distribution, after which $P$ was computed by $P = P^{1/2} P^{1/2}$. The matrix $Q$ was generated randomly from its spectral decomposition with a spectral radius of 10. We chose $c_1 = 1$, $c_2 = c_3 = 0.1$, and $n = 4$. Thus, the smoothness factor $L_r$ is given by $L_r = c_2 = 0.1$. The parameters of backtracking line search were given by $\alpha = 0.3$, $\beta = 0.5$, $\eta_{\text{max}} = 1$, and $\eta_{\text{min}} = 0.005$.

Results The results are shown in Fig. 1, where Fig. 1a and 1b are from the GC algorithm, and Fig. 1c is from the method by Qu et al. [18]. The blue part of the curve in Fig. 1a and 1b represents the model-free regime, and the red represents the model-based regime. To better reflect the actual computational cost, we used the number of function evaluations rather than the number of iterations to quantify the running time of the algorithms. The vertical axis is the error $\|x - x^*\|$, plotted in a logarithmic scale, where $x^*$ is the optimal solution that minimizes $f = \hat{f} + r$. We terminated both algorithms when $\|x - x^*\| < 10^{-3}$. The error $\|x - x^*\|$ was only updated upon the completion of backtracking line search. Since each test of the Armijo condition (Line 2 in Algorithm 2) in
backtracking line search requires one function evaluation, the error \( \|x - x^*\| \) sometimes remained unchanged over multiple consecutive function evaluations when backtracking line search was in progress.

It can be seen that the GC algorithm required fewer function evaluations for achieving the same level of suboptimality. Whether the GC algorithm stays longer in the model-based regime or in the model-free regime depends on the choice of \( \gamma \) in (11), which determines whether Algorithm 3 should continue to stay in the model-based regime. When a larger \( \gamma = 0.9 \) was used, the algorithm stayed for fewer iterations in the model-based regime because condition (11) became more difficult to satisfy.

### 5.2 Modified LQR problem

Consider the modified LQR problem defined by (1) and (2) in Section 2. The problem instance is similar to the one used in [18]. The matrices \( A \) and \( B \) were generated randomly, with each entry drawn from the standard Gaussian distribution. To simplify, the nonlinear error term \( h \) in the dynamics (1) was set to only depend on the system state \( x \). The simplified error term was defined by \( h(x) = \ell x / (1 - 0.9 \sin x) \) with \( \ell = 0.01 \). The quadratic cost in (2) was defined by \( Q = 2I \) and \( R = I \). The dimensions of the problem were set as \( n = 4 \) and \( p = 3 \). (In comparison, \( n = p = 2 \) was used in [18].) The parameters of backtracking line search were given by \( \alpha = 0.3, \beta = 0.5, \eta_{\text{max}} = 1, \) and \( \eta_{\text{min}} = 0.05 \).

Unlike the quadratic functions studied in Section 5.1, in the setting of modified LQR, it is difficult to obtain the value of \( L_r \). Instead, we treated \( L_r \) as a hyperparameter, performed a number of trials with different choices of \( L_r \), and reported the best result.

#### 5.2.1 Computing the gradient

The gradient of the cost \( \hat{C} \) for the linear system can be computed analytically [10] as \( \nabla \hat{C}(K) = 2E_K \Sigma_K \), where \( E_K = (R + B^T P_K B)K - B^T P_K A \) and \( \Sigma_K = \mathbb{E}_{x_0 \sim \mathcal{D}} \sum_{t=0}^{\infty} x_t x_t^T \). The matrix \( P_K \) in the expression for \( E_K \) is the solution to the Lyapunov equation
\[
A_K^T P_K A_K - P_K + Q + K^T R K = 0,
\]
where \( A_K = A - BK \). For the cost \( C \) for the nonlinear system, we computed \( \nabla C(K) \) by a zeroth-order method with \( C \) approximated by a finite-horizon empirical cost
\[
C_{\text{emp}}^T(K) \triangleq \mathbb{E}_{x_0 \sim \mathcal{D}} \left( \sum_{t=0}^{T} x_t^T Q x_t + u_t^T R u_t \right),
\]
where \( \mathcal{D} \) is a uniform distribution over \( n \) distinct values \( x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(n)} \). The procedure for computing \( \nabla C \) is presented in Algorithm 4. In all the experiments, the time horizon \( T \) was set to 50, and the sampling radius \( r \) was set to 0.001.

#### 5.2.2 Results

Fig. 2 compares the result obtained from the GC algorithm and the one from the model-free method by Qu et al. [18]. Because the modified LQR does not admit an optimal solution in closed form, the optimal feedback gain \( K^* \) was computed approximately by running the model-free method in [18] until \( \| \nabla C(K^*) \| < 10^{-4} \). We terminated both algorithms when \( \| K - K^* \| < 10^{-4} \).
**Algorithm 4** Computing the gradient of $C$

**Input:** $K$, $r$, $n$, $p$, $T$, $A$, $B$, $h$, $Q$, $R$

**Output:** $\nabla C(K)$

1. **for** $j = 1, 2, ..., pm$ **do**
2.     // RESHAPE($\cdot$, $p \times n$) converts a $pm$-dimensional vector to a $p \times n$ matrix.
3.     $U_j \leftarrow$ RESHAPE($r \cdot e_j$, $p \times n$), where $e_j$ is a unit basis vector whose $j$-th entry is 1.
4.     **for** $k = 1, 2, ..., n$ **do**
5.         $x_0^{(k)} \leftarrow x_0^{(k)}$
6.         **for** $t = 0, 1, ..., T$ **do**
7.             $u_t \leftarrow -(K + U_j) x_t$
8.             $x_{t+1} = Ax_t + Bu_t + h(x_t)$
9.         **end for**
10.        $C_j^{(k)} \leftarrow \sum_{t=0}^{T} x_t^\top Q x_t + u_t^\top R u_t$
11.       **end for**
12.       $C_j \leftarrow \frac{1}{n} \sum_{k=1}^{n} C_j^{(k)}$
13. **end for**
14. **return** $\nabla C(K) = \frac{1}{d} \sum_{j=1}^{d} \frac{d}{pm} C_j U_j$, where $d = pn$

---

![Comparison between the convergence of (a) the GC algorithm with $\eta_{\text{min}} = 0.05$ and (b) the model-free method in [18] for the modified LQR problem. While the GC used fewer function evaluations, it still spent most of the time in the model-free regime, indicated in blue.](image-url)
Figure 3: Effect of the minimum step size $\eta_{\text{min}}$ on the convergence of the GC algorithm: (a) $\eta_{\text{min}} = 5 \times 10^{-4}$. (b) $\eta_{\text{min}} = 5 \times 10^{-9}$. (c) $\eta_{\text{min}} = 0.5$. Choosing $\eta_{\text{min}}$ either too small or too large increased the total number of function evaluations. (Blue: model-free regime. Red: model-based regime.)

Effect of $\gamma$ and $L_r$ The choices of $\gamma$ and $L_r$ affect condition (11), which is used to monitor whether the sufficient decrease condition holds upon the success of backtracking line search. When the minimum step size $\eta_{\text{min}} = 0.05$ was used, both $\gamma$ and $L_r$ were found to have little effect on the performance of the GC algorithm. For instance, as $\gamma$ varied between 0.0001 to 0.9999, the convergence of the GC algorithm remained the same, as represented in Fig. 2a. Upon inspecting the numerical simulation results, we found that, for the majority of the time, the GC algorithm was found to leave the model-based regime almost immediately after entering. Moreover, the GC algorithm left the model-based regime mostly due to the failure of backtracking line search rather than violating condition (11). Since neither $\gamma$ nor $L_r$ affects backtracking line search, this explains why changing $\gamma$ and $L_r$ did not help the GC algorithm stay in the model-based regime longer so as to reduce the number of function evaluations.

Effect of $\eta_{\text{min}}$ Since backtracking line search was what prevented the GC algorithm from staying in the model-based regime, we changed the minimum step size $\eta_{\text{min}}$ from 0.05 to $5 \times 10^{-4}$. The result is shown in Fig. 3a from which it can be seen that the algorithm stayed longer in the model-based regime, and the total number of function evaluations was smaller compared with Fig. 2a. However, upon further decreasing $\eta_{\text{min}}$ to $5 \times 10^{-9}$, the total number of function evaluations increased, as shown in Fig. 3b. Recall that each attempt within backtracking line search decreases the previous step size by a constant factor $\beta$ and uses one function evaluation to determine whether the new step size is appropriate. For a smaller $\eta_{\text{min}}$, in order to declare the failure of line search, backtracking line search requires more attempts and hence more function evaluations before the proposed step size $\eta$ reaches $\eta_{\text{min}}$. To reduce the total number of iterations for reaching a desired level of suboptimality, one should choose $\eta_{\text{min}}$ to balance between the number of iterations and the number of function evaluations (per iteration) in the model-based regime. Choosing a smaller $\eta_{\text{min}}$ will make the algorithm stay in the model-based regime for more iterations and hence better exploit model information at the expense of more function evaluations per iteration.

We also experimented with a larger value of $\eta_{\text{min}}$ by choosing $\eta_{\text{min}} = 0.5$, the result of which is shown in Fig. 3c. As expected, a larger $\eta_{\text{min}}$ caused the algorithm to spend fewer iterations in the model-based regime because the line search termination condition was tightened. In fact, for $\eta_{\text{min}} = 0.5$, the GC algorithm did not enter the model-based regime at all; the algorithm only stayed in the model-free regime and behaved identically to the model-free method in 18 shown in Fig. 2b.
6 Conclusion

We have proposed a framework that combines model-based and model-free methods for optimal control. The framework formulates the optimal control problem as a composite optimization problem whose objective function is described by a sum of two terms: a model-based term whose explicit expression can be obtained from an approximate model of the system dynamics and a residual term that captures the unknown modeling error. To solve the composite optimization problem, we have developed a hybrid gradient-based algorithm that uses gradient information from the approximate model with compensation from intermittent model-free updates. Both theoretical analysis and numerical experiments show that the algorithm uses fewer function evaluations than a purely model-free policy gradient algorithm for reaching the same level of suboptimality.

References

[1] Pieter Abbeel, Morgan Quigley, and Andrew Y. Ng. Using inaccurate models in reinforcement learning. In Proceedings of the 23rd International Conference on Machine Learning - ICML '06, pages 1–8, Pittsburgh, Pennsylvania, 2006. ACM Press.

[2] C.G. Atkeson and J.C. Santamaria. A comparison of direct and model-based reinforcement learning. In Proceedings of International Conference on Robotics and Automation, volume 4, pages 3557–3564 vol.4, April 1997.

[3] Amir Beck and Marc Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. SIAM Journal on Imaging Sciences, 2(1):183–202, January 2009.

[4] Stephen P. Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, Cambridge, UK ; New York, 2004.

[5] Jingjing Bu, Afshin Mesbahi, Maryam Fazel, and Mehran Mesbahi. LQR through the Lens of First Order Methods: Discrete-time Case. arXiv:1907.08921 [cs, eess, math], July 2019.

[6] Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning, pages 703–711. PMLR, July 2017.

[7] Xin Chen, Guannan Qu, Yujie Tang, Steven Low, and Na Li. Reinforcement Learning for Decision-Making and Control in Power Systems: Tutorial, Review, and Vision. arXiv:2102.01168 [cs, eess], 2021.

[8] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. Introduction to Derivative-Free Optimization. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, January 2009.

[9] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian Processes for Data-Efficient Learning in Robotics and Control. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(2):408–423, February 2015.

[10] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator. In Proceedings of the 35th International Conference on Machine Learning, pages 1467–1476. PMLR, July 2018.
[11] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Łojasiewicz Condition. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 9851, pages 795–811. Springer International Publishing, Cham, 2016.

[12] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, volume 3, pages 2619–2624 Vol.3, April 2004.

[13] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[14] A. S. Lewis and S. J. Wright. A proximal method for composite minimization. *Mathematical Programming*, 158(1-2):501–546, 2016.

[15] Yu. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[16] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd ed edition, 2006.

[17] Neal Parikh and Stephen Boyd. Proximal Algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, January 2014.

[18] Guannan Qu, Chenkai Yu, Steven Low, and Adam Wierman. Combining Model-Based and Model-Free Methods for Nonlinear Control: A Provably Convergent Policy Gradient Approach. *arXiv:2006.07476 [cs, eess, math]*, June 2020.

[19] Benjamin Recht. A Tour of Reinforcement Learning: The View from Continuous Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

[20] R. Tyrrell Rockafellar. Monotone Operators and the Proximal Point Algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, August 1976.

[21] Shirli Di-Castro Shashua, Dotan Di Castro, and Shie Mannor. Sim and Real: Better Together. In *Thirty-Fifth Conference on Neural Information Processing Systems*, May 2021.

[22] Stephen Tu and Benjamin Recht. The Gap Between Model-Based and Model-Free Methods on the Linear Quadratic Regulator: An Asymptotic Viewpoint. In *Proceedings of the Thirty-Second Conference on Learning Theory*, pages 3036–3083. PMLR, June 2019.

[23] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M. Bayen. Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control. *arXiv:1710.05465 [cs]*, 2017.