Iterative Linearized Control: Stable Algorithms and Complexity Guarantees

Vincent Roulet $^1$ Siddhartha Srinivasa $^2$ Dmitriy Drusvyatskiy $^3$ Zaid Harchaoui $^1$

Abstract

We examine popular gradient-based algorithms for nonlinear control in the light of the modern complexity analysis of first-order optimization algorithms. The examination reveals that the complexity bounds can be clearly stated in terms of calls to a computational oracle related to dynamic programming and implementable by gradient back-propagation using machine learning software libraries such as PyTorch or TensorFlow. Finally, we propose a regularized Gauss-Newton algorithm enjoying worst-case complexity bounds and improved convergence behavior in practice. The software library based on PyTorch is publicly available.

Introduction

Finite horizon discrete time nonlinear control has been studied for decades, with applications ranging from spacecraft dynamics to robot learning (Bellman, 1971; Bertsekas, 2005). Popular nonlinear control algorithms, such as differential dynamic programming or iterative linear quadratic Gaussian algorithms, are commonly derived using a linearization argument relating the nonlinear control problem to a linear control problem (Todorov & Li, 2003; Li & Todorov, 2007).

We examine nonlinear control algorithms based on iterative linearization techniques in the light of the modern complexity analysis of first-order optimization algorithms. We first reformulate the problem as the minimization of an objective that is written as a composition of smooth functions. Owing to this reformulation, we can frame several popular nonlinear control algorithms as first-order optimization algorithms applied to this objective.

Related work. Differential dynamic programming (DDP) and iterative linearization algorithms are popular algorithms for finite horizon discrete time nonlinear control (Tassa et al., 2014). DDP is based on approximating the Bellman equation at the current trajectory in order to use standard dynamic programming. Up to our knowledge, the complexity analysis of DDP has been limited; see (Mayne, 1966; Jacobson & Mayne, 1970; Todorov & Li, 2003) for classical analyses of DDP.

Iterative linearization algorithms such as the iterative linear quadratic regulator (ILQR) or the iterative linearized Gaussian algorithm (ILQG) linearize the trajectory in order to use standard dynamic programming (Li & Todorov, 2004; 2007). Again, the complexity analysis of ILQG for instance has been limited. It is worthwhile to mention related approaches in the non-linear model predictive control area (Grüne & Pannek, 2017; Richter et al., 2012; Dontchev et al., 2018).
We adopt the point of view of the complexity theory of first-order optimization algorithms. The exact form of a Newton step and Gauss-Newton step in a nonlinear control problem is well-known; see Dunn & Bertsekas (1989); Sideris & Bobrow (2005). However, while the importance of the addition of a proximal term is now well-understood, several algorithms involving such steps have not been revisited yet, such as ILQR (Liao & Shoemaker, 1992; Li & Todorov, 2004). Our work shows how to make these improvements.

We also show how gradient back-propagation, i.e., automatic differentiation (Griewank & Walther, 2008), a popular technique usually derived using either a chain rule argument or a Lagrangian framework (Bertsekas, 2005; LeCun et al., 1988), allows one to solve the dynamic programming problems arising in linear quadratic control. Consequently, the subproblems that arise when using iterative linearization for nonlinear control can be solved with calls to an automatic differentiation oracle implementable in PyTorch or TensorFlow (Abadi et al., 2015; Paszke et al., 2017; Kakade & Lee, 2018).

The regularized Gauss-Newton method was extensively studied to minimize the nonlinear least squares objective function,
\[ f(x) = \frac{1}{2} \sum_{i=1}^{N} (y_i - \phi_i(x))^2, \]
through
\[ x_{t+1} = \phi_t(x_t), \]
starting from a given \( x_0 \), until a horizon \( T \), where the functions \( \phi_t : \mathbb{R}^d \times \mathbb{R}^p \rightarrow \mathbb{R}^d \) are assumed to be continuously differentiable. Optimality is measured through the convex costs \( h_t, g_t \), on the state and control variables \( x_t, u_t \) respectively, defining the discrete time quadratic control problem
\[
\begin{align*}
\min_{x_0, \ldots, x_T, u_0, \ldots, u_{T-1} \in \mathbb{R}^p} & \sum_{t=1}^{T} h_t(x_t) + \sum_{t=0}^{T-1} g_t(u_t) \\
\text{subject to} & \quad x_{t+1} = \phi_t(x_t, u_t), \quad x_0 = \hat{x}_0,
\end{align*}
\]
where, here and thereafter, the dynamics must be satisfied for \( t = 0, \ldots, T - 1 \).

**Noisy dynamics.** The discrepancy between the model and the dynamics can be taken into account by considering noisy dynamics as
\[
x_{t+1} = \phi_t(x_t, u_t, w_t),
\]
where \( w_t \sim \mathcal{N}(0, I_p) \) for \( t = 0, \ldots, T - 1 \). The resulting discrete time control problem consists of optimizing the average cost under the noise \( \hat{w} = (w_0; \ldots; w_{T-1}) \) as
\[
\begin{align*}
\min_{x_0, \ldots, x_T, u_0, \ldots, u_{T-1} \in \mathbb{R}^p} & \mathbb{E}_{\hat{w}} \left[ \sum_{t=1}^{T} h_t(x_t) + \sum_{t=0}^{T-1} g_t(u_t) \right] \\
\text{subject to} & \quad x_{t+1} = \phi_t(x_t, u_t, w_t), \quad x_0 = \hat{x}_0.
\end{align*}
\]

**Costs and penalties.** The costs on the trajectory can be used to force the states to follow a given orbit \( \hat{x}_1, \ldots, \hat{x}_T \) as
\[
h_t(x_t) = \frac{1}{2} (x_t - \hat{x}_t)^\top Q_t (x_t - \hat{x}_t), \quad \text{with} \quad Q_t \succeq 0,
\]
which gives a quadratic tracking problem, while the regularization penalties on the control variables are typically quadratic functions
\[
g_t(u_t) = \frac{1}{2} u_t^\top R_t u_t, \quad \text{with} \quad R_t \succ 0.
\]
The regularization penalties can also encode constraints on the control variable such as the indicator function of a box
\[
g_t(u_t) = \ell_{\{u; c^- \leq u \leq c^+\}}(u_t), \quad \text{with} \quad c^-_t, c^+_t \in \mathbb{R}^p,
\]
where \( \ell_S \) denotes the indicator function of a set \( S \).

**Iterative Linear Control algorithms.** We are interested in the complexity analysis of algorithms such as the iterative linear quadratic regulator (ILQR) algorithm of Li & Todorov (2007), used for exact dynamics, which iteratively computes the solution of
\[
\begin{align*}
\min_{y_0, \ldots, y_T \in \mathbb{R}^d} & \sum_{t=1}^{T} q_t(x_t^{(k)}+y_t) + \sum_{t=0}^{T-1} q_t(u_t^{(k)}+v_t) \\
\text{subject to} & \quad y_{t+1} = \ell_{\phi_t}(y_t, v_t), \quad y_0 = 0,
\end{align*}
\]
where $\bar{u}^{(k)}$ is the current command, $\tilde{u}^{(k)}$ is the corresponding trajectory given by (1), $q_{h}, q_{g}$ are quadratic approximations of the costs $h_{t}, g_{t}$ around respectively $x_{t}^{(k)}, u_{t}^{(k)}$ and $\ell_{\phi_{t}}$ is the linearization of $\phi_{t}$ around $(x_{t}^{(k)}, u_{t}^{(k)})$. The next iterate is then given by $\bar{u}^{(k+1)} = \bar{u}^{(k)} + 2\nu^{*}$ where $\nu^{*}$ is the solution of (8) and $\nu$ is a step-size given by a line-search method. To understand this approach, we frame the problem as the minimization of a composition of functions.

Formulation as a composite optimization problem. We call an optimization problem a composite optimization problem if it consists in the minimization of a composition of functions. For a fixed command $\bar{u}$ in $\mathbb{R}^m$, denote by $\bar{x}(\bar{u}) = (\bar{x}_{1}(\bar{u}); \ldots ; \bar{x}_{r}(\bar{u})) \in \mathbb{R}^d$ the trajectory given by the exact dynamics

$$\bar{x}_{1}(\bar{u}) = \phi_{0}(\bar{x}_{0}, u_{t}), \quad \bar{x}_{t+1}(\bar{u}) = \phi_{t}(\bar{x}_{t}(\bar{u}), u_{t}).$$

Similarly denote by $\tilde{x}(\bar{u}, \bar{v})$ the trajectory in the noisy case. Denoting the total cost by $h(\bar{x}) = \sum_{t=0}^{\tau} h_{t}(x_{t})$, the total penalty by $g(\bar{u}) = \sum_{t=0}^{\tau} g_{t}(u_{t})$, the discrete time control problem (2) with exact dynamics reads

$$\min_{\bar{u} \in \mathbb{R}^m} f(\bar{u}) \triangleq h(\bar{x}(\bar{u}))+g(\bar{u}),$$

and with noisy dynamics,

$$\min_{\bar{u} \in \mathbb{R}^m} f(\bar{u}) \triangleq \mathbb{E} \left[ h(\bar{x}(\bar{u}, \bar{v})) \right] + g(\bar{u}),$$

i.e., we obtain a composite optimization problem whose structure can be exploited to derive oracles on the objective.

2. Oracles in discrete time control

We adopt here the viewpoint of the complexity theory of first-order optimization. Given the composite problem (10), what are the relevant oracles and what are the complexities of calls to these oracles? We highlight how classical algorithms in discrete time control can be seen as standard optimization algorithms. We first consider exact dynamics $\phi_{t}$ of the form (1) and unconstrained cost penalties such as (6).

2.1. Exact and unconstrained setting

Model minimization. Each step of the optimization algorithm is defined by the minimization of a regularized model of the objective. For example, a gradient step on a point $\bar{u}$ with step-size $\gamma$ corresponds to linearizing both $h$ and $x$ and defining the linear model

$$\ell_{f}(\bar{u} + \bar{v} ; \bar{u}) = \ell_{h}(\bar{x}^{(\bar{u})} + \nabla \bar{x}^{(\bar{u})} \bar{v} ; \bar{x}^{(\bar{u})}) + \ell_{g}(\bar{u} + \bar{v} ; \bar{u})$$

of the objective $f$, where $\ell_{h}(\bar{x} + \bar{y} ; \bar{x}) = h(\bar{x}) + \nabla h(\bar{x})^\top \bar{y}$ and $\ell_{g}(\bar{u} + \bar{v} ; \bar{u})$ is defined similarly. Then, this model with a proximal regularization is minimized in order to get the next iterate

$$\bar{u}^{+} = \bar{u} + \arg\min_{\bar{v} \in \mathbb{R}^m} \left\{ \ell_{f}(\bar{u} + \bar{v} ; \bar{u}) + \frac{1}{2\gamma} \| \bar{v} \|_{2}^{2} \right\}.$$  

Different models can be defined to better approximate the objective. For example, if only the mapping $\tilde{x}$ is linearized, this corresponds to defining the convex model at a point $\bar{u}$

$$c_{f}(\bar{u} + \bar{v} ; \bar{u}) = h(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v} ; \bar{x}(\bar{u})) + g(\bar{u} + \bar{v}).$$

We get then a regularized Gauss-Newton step on a point $\bar{u} \in \mathbb{R}^m$ with step size $\gamma > 0$ as

$$\bar{u}^{+} = \bar{u} + \arg\min_{\bar{v} \in \mathbb{R}^m} \left\{ c_{f}(\bar{u} + \bar{v} ; \bar{u}) + \frac{1}{2\gamma} \| \bar{v} \|_{2}^{2} \right\}.$$  

Although this model better approximates the objective, its minimization may be computationally expensive for general functions $h$ and $g$. We can use a quadratic approximation of $h$ around the current mapping $\tilde{x}(\bar{u})$ and linearize the trajectory around $\bar{u}$ which defines the quadratic model

$$q_{f}(\bar{u} + \bar{v} ; \bar{u}) = q_{h}(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v} ; \tilde{x}(\bar{u})) + q_{g}(\bar{u} + \bar{v} ; \bar{u}),$$

where $q_{h}(\tilde{x} + \bar{y} ; \tilde{x}) \triangleq h(\tilde{x}) + \nabla h(\tilde{x})^\top \bar{y} + \bar{y}^\top \nabla^{2} h(\tilde{x}) \bar{y}/2$ and $q_{g}(\bar{u} + \bar{v} ; \bar{u})$ is defined similarly. A Levenberg-Marquardt step with step-size $\gamma$ consists in minimizing the model (15) with a proximal regularization

$$\bar{u}^{+} = \bar{u} + \arg\min_{\bar{v} \in \mathbb{R}^m} \left\{ q_{f}(\bar{u} + \bar{v} ; \bar{u}) + \frac{1}{2\gamma} \| \bar{v} \|_{2}^{2} \right\}.$$  

Model-minimization steps by linear optimal control. Though the chain rule gives an analytic form of the gradient, we can use the definition of a gradient step as an optimization sub-problem to understand its implementation. Formally, the above steps (12), (14), (16), define a model of the discrete time control objective $f$ in (10) on a point $\bar{u}$

$$m_{f}(\bar{u} + \bar{v} ; \bar{u}) = m_{h}(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v} ; \tilde{x}(\bar{u})) + m_{g}(\bar{u} + \bar{v} ; \bar{u}),$$

where $m_{h} = \sum_{t=1}^{\tau} m_{h_{t}}$, $m_{g} = \sum_{t=0}^{\tau-1} m_{g_{t}}$ are models of $h$ and $g$ respectively, composed of models on the individual variables. The model-minimization step with step-size $\gamma$,

$$\bar{u}^{+} = \bar{u} + \arg\min_{\bar{v} \in \mathbb{R}^m} \left\{ m_{f}(\bar{u} + \bar{v} ; \bar{u}) + \frac{1}{2\gamma} \| \bar{v} \|_{2}^{2} \right\},$$

amounts then to a linear control problem as shown in the following proposition.
Proposition 2.1. The model-minimization step (17) for discrete time control problem (2) written as (10) is given by 
\[
\hat{u}^* = \hat{u} + \tilde{v}^* \quad \text{where} \quad \tilde{v}^* = (v_0^*; \ldots; v_{\tau-1}^*) \quad \text{is the solution of}
\]
\[
\min_{y_0, \ldots, y_\tau \in \mathbb{R}^p} \sum_{t=0}^{\tau} m_{h_t}(x_t + y_t; x_t) + \sum_{t=0}^{\tau-1} m_{g_t}(u_t + v_t; u_t)
\]
subject to \quad \gamma_t = \Phi_{t,x} y_t + \Phi_{t,u} v_t, \quad y_0 = 0, \quad (18)

where \(\gamma_t\) is the solution of \(\Phi_t(x) = \nabla x \phi_t(x_t, u_t), \Phi_t(u) = \nabla u \phi_t(x_t, u_t)\) and \(x_t = \bar{x}_t(\hat{u})\).

Proof. Recall that the trajectory defined by \(\hat{u}\) reads
\[
\bar{x}_1(\hat{u}) = \phi_0(\bar{x}_0, F_0^\top \hat{u}), \quad \bar{x}_{t+1}(\hat{u}) = \phi_t(\bar{x}_t(\hat{u}), F_t^\top \hat{u}),
\]
where \(F_t = e_t+1 \otimes I_p \in \mathbb{R}^{p \times p}, e_t \in \mathbb{R}^p\) is the \(t\)th canonical vector in \(\mathbb{R}^p\), such that \(F_t^\top \hat{u} = u_t\). The gradient reads
\[
\nabla \bar{x}_1(\hat{u}) = F_0 \nabla u \phi_0(x_0, u_0) \quad \text{followed by}
\]
\[
\nabla \bar{x}_{t+1}(\hat{u}) = \nabla \bar{x}_t(\hat{u}) \nabla x \phi_t(x_t, u_t) + F_t \nabla u \phi_t(x_t, u_t),
\]
where \(x_t = \bar{x}_t(\hat{u})\) and \(x_0 = \bar{x}_0\). For a given \(\bar{v} = (v_0; \ldots; v_{\tau-1})\), the product \(\bar{y} = (y_1; \ldots; y_\tau) = \nabla \bar{x}(\hat{u})^\top \bar{v}\) reads
\[
y_1 = \nabla u \phi_0(x_0, u_0) \quad \text{followed by}
\]
\[
y_{t+1} = \nabla x \phi_t(x_t, u_t)^\top y_t + \nabla u \phi_t(x_t, u_t)^\top v_t,
\]
where we used that \(y_t = \nabla \bar{x}_t(\hat{u})^\top \bar{v}\). Plugging this into (17) gives the result.

Dynamic programming. If the models used in (17) are linear or quadratic in the states and quadratic in the controls, the resulting linear control problems (18) can be solved efficiently using dynamic programming, i.e., with a linear cost in \(\tau\), as presented in the following proposition. The cost is \(O(\tau p^2d^3)\). Details on the implementation for quadratic costs are provided in (Roulet et al., 2019).

Since the leading dimension of the discrete time control problem is the length of the trajectory \(\tau\), all of the above optimization steps have roughly the same cost. This means that, in discrete time control problems, second order steps such as (16) are roughly as expensive as gradient steps.

Proposition 2.2. Model-minimization steps of the form (17) for discrete time control problem (2) written as (10) with linear or quadratic convex models \(m_h\) and \(m_g\) can be solved in linear time with respect to the length of the trajectory \(\tau\) by dynamic programming.

The proof of the proposition relies on the dynamic programming approach explained below. The linear optimal control problem (18) can be divided into smaller subproblems and then solved recursively. Consider the linear optimal control problem (18) as
\[
\min_{y_0, \ldots, y_\tau \in \mathbb{R}^p} \sum_{t=0}^{\tau} q_{h_t}(y_t) + \sum_{t=0}^{\tau-1} q_{g_t}(v_t)
\]
subject to \(y_{t+1} = \ell_t(y_t, v_t), \quad y_0 = 0, \quad (19)\)

where \(\ell_t\) is a linear dynamic in state and control variables, \(q_{g_t}\) are strictly convex quadratics and \(q_{h_t}\) are convex quadratic or linear functions. For \(0 \leq t \leq \tau\), given \(\hat{y}_t\), define the cost-to-go from \(\hat{y}_t\), as the solution of
\[
\hat{y}_{t+1} = \min_{v_{t+1}} \sum_{t'=t}^{\tau} q_{h_{t'}}(y_{t'}) + \sum_{t'=t}^{\tau-1} q_{g_{t'}}(v_{t'}) \quad (20)
\]
solved for \(v^*_{t+1}(\hat{y}_t) = \arg \min_{v_{t+1}} \left\{ q_{h_{t'}}(y_{t'}) + c_{t+1}(\ell_t(\hat{y}_t, v_{t'})) \right\} \quad (21)\).

The cost-to-go functions can be computed recursively by the Bellman equation for \(t \in \{\tau - 1, \ldots, 0\}\),
\[
c_t(\hat{y}_t) = q_{h_t}(\hat{y}_t) + \min_{v_{t+1}} \left\{ q_{g_{t+1}}(v_{t+1}) + c_{t+1}(\ell_t(\hat{y}_t, v_{t+1})) \right\} \quad (21)
\]
solved for \(v^*_t(\hat{y}_t) = \arg \min_{v_{t+1}} \left\{ q_{g_{t+1}}(v_{t+1}) + c_{t+1}(\ell_t(\hat{y}_t, v_{t+1})) \right\} \).

The final cost initializing the recursion is defined as \(c_{\tau}(\bar{y}_\tau) = q_{h_\tau}(\bar{y}_\tau)\). For quadratic costs and linear dynamics, the problems defined in (21) are themselves quadratic problems that can be solved analytically to get an expression for \(c_t\).

The solution of (19) is given by computing \(c_0(0)\), which amounts to iteratively solving the Bellman equations starting from \(\hat{y}_0 = 0\), i.e., getting the linear optimal control at the given state and moving along the dynamics to define the next cost-to-go function to minimize
\[
v^*_{t+1}(\hat{y}_t), \quad y_{t+1} = \ell_t(y_t, v^*_{t+1}). \quad (22)\]

The cost of the overall dynamic procedure that involves a backward pass to compute the cost-to-go functions and a roll-out pass to compute the optimal controls is therefore linear in the length of the trajectory \(\tau\). The main costs lie in solving linear or quadratic problems in the Bellman equation (21) which only depend on the state and control dimensions \(d\) and \(p\).

Back-propagation as dynamic programming. We illustrate the derivations for a gradient step in the following proposition that shows a cost of \(O(\tau(pd + d^2))\). We recover the well-known gradient back-propagation algorithm used to compute the gradient of the objective. The dynamic programming viewpoint provides here a natural derivation.

Proposition 2.3. A gradient step (12) for discrete time control problem (2) written as (10) and solved by dynamic programming amounts to
1. A forward pass that computes the derivatives $\nabla_x \phi_t(x_t, u_t)$, $\nabla_u \phi_t(x_t, u_t)$, $\nabla h_t(x_t)$, $\nabla g_t(u_t)$ for $t = 0, \ldots, \tau$ along the trajectory $x_{t+1} = \phi_t(x_t, u_t)$.

2. A backward pass that computes linear cost-to-go functions as $c_t(y_t) = \lambda^T_t y_t + \mu_t$ where $\lambda^T = \nabla h_t(x_t)$, $\mu_t = \nabla h_t(x_t) + \nabla g_t(x_t) \lambda_{t+1}$, for $t = \tau - 1, \ldots, 0$.

3. An update pass that outputs $u^*_t = u_t - \gamma (\nabla_u \phi_t(x_t, u_t) \lambda_t + \nabla g_t(u_t))$, for $t = 0, \ldots, \tau - 1$.

2.2. Noisy or constrained settings

**Noisy dynamics.** For inexact dynamics defining the problem (11), we consider a Gaussian approximation of the linearized trajectory around the exact current trajectory. Formally, the Gaussian approximation of the random linearized trajectory $\tilde{y}(\tilde{u} + \bar{v}, \bar{w}) = \tilde{x}(\tilde{u}, \bar{w}) + \nabla_{\bar{w}} \tilde{x}(\tilde{u}, \bar{w})^T \bar{v}$ around the exact linearized trajectory given for $\bar{w} = 0$ reads

$$
\tilde{y}(\tilde{u} + \bar{v}, \bar{w}) = \tilde{x}(\tilde{u}, 0) + \nabla_{\bar{w}} \tilde{x}(\tilde{u}, 0)^T \bar{v} + \nabla_{\bar{w}}^2 \tilde{x}(\tilde{u}, 0)[\bar{v}, \tilde{u}, \tilde{w}],
$$

which satisfies $\mathbb{E}_{\bar{w}}[\tilde{y}(\tilde{u}+\bar{v}; \bar{u}, \bar{w})] = \tilde{x}(\tilde{u}, 0) + \nabla_{\bar{w}} \tilde{x}(\tilde{u}, 0)^T \bar{v}$, see (Roulet et al., 2019) for gradient and tensor notations.

The quadratic model we consider for the state cost is then of the form

$$
\mathbb{E}_{\bar{w}} \left[ q_h \left( \tilde{x}(\tilde{u} + \bar{v}; \tilde{u}, \bar{w}); \tilde{x}(\tilde{u}, 0) \right) \right].
$$

(23)

For simple dynamics $\phi_t$, their minimization with an additional proximal term amounts to a linear quadratic Gaussian control problem as stated in the following proposition.

**Proposition 2.4.** Assume $\nabla^2_x \phi_t$, $\nabla^2_{xu} \phi_t$ and $\nabla^2_{uu} \phi_t$ to be zero. The model minimization step (17) for model (23) is given by $\bar{u} = \bar{v} + \bar{w}$ where $\bar{w}$ is the solution of

$$
\min_{y, \bar{w}} \sum_{t=1}^{\tau} \mathbb{E}_{\bar{w}} \left[ q_h \left( x_t + \bar{v} + y_t; x_t \right) \right] + \sum_{t=0}^{\tau-1} q_g \left( u_{t+1} + v_t + t \right),
$$

s.t. $y_{t+1} = \Phi_t x_t + \Phi_t^T v_t + \Phi_t^T \bar{w} + \Phi_t w_t, \Phi_t w_t = \nabla_{\bar{w}} \phi_t(x_t, u_t, 0), \Phi_t w_t = \nabla_{xu} \phi_t(x_t, u_t, 0), \Phi_t w_t = \nabla_{uu} \phi_t(x_t, u_t, 0), \Phi_t w_t = \tilde{x}_t(\tilde{u}, 0)$, and $q_g = g_t + (2\gamma)^{-1} \| u_t - u_t^* \|^2$.

(24)

The linear control problem (24) can again be solved by dynamic programming by modifying the Bellman equation (21) in the backward pass, i.e., by solving adaptively for white noise $w_t$,

$$
c_t(\tilde{y}_t) = q_t(\tilde{y}_t) + \min_{v_t} q_t(\tilde{v}_t) + \mathbb{E}_{w_t} \left[ c_{t+1}(\ell_t(\tilde{v}_t, v_t, w_t)) \right].
$$

**Dealing with constraints.** For constrained control problems with exact dynamics, the model-minimization steps will amount to linear control problems under constraints, which cannot be solved directly by dynamic programming. However, their resolution by an interior point method boils down to solving linear quadratic control problems each of which has a low computational cost as shown before.

Formally, the resulting subproblems we are interested in are linear quadratic control problems under constraints of the form

$$
\min_{y_0, \ldots, y_{\tau-1} \in \mathbb{R}^d} \sum_{t=1}^{\tau} q_h(y_t) + \sum_{t=0}^{\tau-1} q_g(v_t)
$$

subject to $y_{t+1} = \ell_t(y_t, v_t), \quad y_0 = 0, \quad v_t \in U_t$,

where $U_t = \{ u : C_t u \leq d_t \}$, $q_t$ are convex quadratics, $g_t$ are strictly convex quadratics and $\ell_t$ are linear dynamics. Interior point methods introduce a log-barrier function $B_k(u) = \log(d_t - C_t u)$ and minimize

$$
\min_{y_0, \ldots, y_{\tau-1} \in \mathbb{R}^p} \sum_{t=1}^{\tau} q_h(y_t) + \sum_{t=0}^{\tau-1} q_g(v_t) + \mu_k B_k(v_t)
$$

subject to $y_{t+1} = \ell_t(y_t, v_t), \quad y_0 = 0$,

where $\mu_k$ increases along the iterates $k$ of the interior point method. We leave the exploration of constrained problems to future work.

3. Automatic-differentiation oracle

We now focus on problems where the cost depends only on the last state. These problems arise when only the final target is defined. See (Roulet et al., 2019) for discussion. They also arise in optimization problems involving successive compositions of functional transformations of a given input as

$$
\min_{u_0, \ldots, u_{\tau-1}} h(\phi_{\tau-2}(\ldots \phi_0(\dot{x}_0, u_0) \ldots, u_{\tau-2}), u_{\tau-1})).
$$

We formalize this class of composite functions in (Roulet et al., 2019). We summarize now the complexity of oracles for problems of the form

$$
\min_{u \in \mathbb{R}^p} h(\xi(\tilde{u})) + g(\tilde{u}),
$$

where $h : \mathbb{R}^d \rightarrow \mathbb{R}, g : \mathbb{R}^p \rightarrow \mathbb{R}$ are convex twice differentiable and $\xi : \mathbb{R}^p \rightarrow \mathbb{R}^d$ is defined as $\xi(\tilde{u}) = \tilde{x}_1(\tilde{u})$ where

$$
\tilde{x}_1(\tilde{u}) = \phi_0(\dot{x}_0, u_0), \quad \tilde{x}_{t+1}(\tilde{u}) = \phi_t(\tilde{x}_t(\tilde{u}), u_t),
$$

(27)

with $\dot{x}_0 \in \mathbb{R}^d$ and continuously differentiable functions $\phi_t : \mathbb{R}^d \times \mathbb{R}^p \rightarrow \mathbb{R}$. 
We can take advantage of automatic-differentiation procedures for this class of problems. An automatic-differentiation procedure can compute any gradient vector product of the form $\nabla \xi(u)z$ for $z \in \mathbb{R}^d$. In our setting, this amounts to solving $\min_{v \in \mathbb{R}^p} -z^T \nabla \xi(\bar{u})^T v + \frac{1}{2} \|v\|^2$ by dynamic programming. We therefore identify calls to a target function oracle as calls to an automatic-differentiation procedure.

**Definition 3.1 (Automatic-differentiation oracle).** An automatic-differentiation oracle is any procedure that computes $\nabla \xi(\bar{u})z$ for $\xi : \mathbb{R}^p \to \mathbb{R}^d$ defined by successive compositions as in (27), $\bar{u} \in \mathbb{R}^p$ and $z \in \mathbb{R}^d$.

Derivatives of the gradient vector product can then be computed at twice the cost of computing the gradient vector product. See (Roulet et al., 2019) for the proof.

**Lemma 3.2.** For a function $\xi : \mathbb{R}^p \to \mathbb{R}^d$ defined by successive compositions as in (27), an automatic-differentiation oracle and a differentiable function $f : \mathbb{R}^p \to \mathbb{R}$, the derivative of $z \mapsto f(\nabla \xi(\bar{u})z)$ for $\bar{u} \in \mathbb{R}^p$ can be computed at twice the cost of computing $\nabla \xi(\bar{u})z$.

We now detail the feasibility and the complexity of the inner-steps of the steps defined in Sec. 2 in terms of automatic-differentiation oracles defined above. The total complexity of the algorithms, when available, is detailed in (Roulet et al., 2019).

**Gradient step.** For a problem of the form (26), a gradient step amounts to computing $\nabla \xi(\bar{u})\nabla h(\xi(\bar{u}))$ and $\nabla g(\bar{u})$ i.e. a single call to the automatic-differentiation oracle.

**Regularized Gauss-Newton step.** For any problem of the form (26), the regularized Gauss-Newton step (14) amounts to solving

$$\min_{\bar{v} \in \mathbb{R}^p} q_h(\xi(\bar{u}) + \nabla \xi(\bar{u})^T \bar{v}) + g(\bar{u} + \bar{v}) + \frac{1}{2\gamma} \|\bar{v}\|^2.$$  

(28)

For smooth objectives $h$ and $g$ and Lipschitz continuous dynamics $\phi_t$, this is a smooth strongly convex problem that can be solved approximately by a linearly convergent method, leading to the inexact regularized Gauss-Newton procedures described by Nesterov (2007); Drusvyatskiy & Paquette (2018). The overall cost of an approximated regularized Gauss-Newton step is then given by the following proposition.

**Proposition 3.3.** For any problem of the form (26), given smooth objectives $h$ and $g$ and smooth and Lipschitz continuous dynamics $\phi_t$, a regularized Gauss-Newton step given by (28) is solved up to $\varepsilon$ accuracy by a fast gradient method with at most

$$O\left(\sqrt{L_h M^2 \bar{\gamma} + L_g \bar{\gamma}} + 1 \log(\varepsilon)\right),$$

calls to an automatic-differentiation oracle, where $M_\xi$ is the Lipschitz continuity constant of $\xi$, $L_h$ and $L_g$ are smoothness constants of respectively $h$ and $g$.

**Levenberg-Marquardt step.** For any problem of the form (26), the Levenberg-Marquardt step (16) amounts to solving

$$\min_{\bar{v} \in \mathbb{R}^p} q_h(\xi(\bar{u}) + \nabla \xi(\bar{u})^T \bar{v}; \xi(\bar{u})) + \frac{1}{2\gamma} \|\bar{v}\|^2,$$  

(29)

where $q_h$ and $q_g$ are quadratic approximations of $h$ and $g$ respectively, assumed to be twice differentiable. Here, duality offers a fast resolution of the step as shown in the following proposition. It shows that the cost of the step (29) is only $2d + 1$ times more than that of a gradient step. Recall also that for quadratic $h$, $g$, the Levenberg-Marquardt step amounts to a regularized Gauss-Newton step.

**Proposition 3.4.** For any problem of the form (26), a Levenberg-Marquardt step (29) is solved exactly with at most $2d + 1$ calls to an automatic-differentiation oracle.

4. Composite optimization

Before analyzing the methods of choice for composite optimization, we review classical algorithms for nonlinear control and highlight improvements for better convergence behavior. All algorithms are completely detailed in (Roulet et al., 2019).

**Differential Dynamic Programming.** Differential Dynamic Programming (DDP) is an algorithm motivated as a dynamic programming procedure applied to a second-order approximation of the Bellman equation. Formally at a given command $\bar{u}$ with associated trajectory $\bar{x} = \bar{x}($\bar{u}$$)$, DDP consists in approximating the cost-to-go functions as

$$c_t(y) = q_h(x_t + y; x_t) + \min_u q_{g_t}(u_t + v; u_t) + q_{c_{t+1} \phi_t}(x_t + y, u_t + v; x_t, u_t),$$

where for a function $f$, $q_f(x + y; x)$ denotes its second order approximation around $x$. We detail the interpretation of DDP as optimization on the state variables in (Roulet et al., 2019).

**ILQR as Gauss-Newton.** DDP was superseded by the Iterative Linearized Quadratic Regulator (ILQR) method of Li & Todorov (2004), presented in Sec. 1. In the case of noisy dynamics, the Linear Quadratic Regulator problem (8) is replaced by a Linear Quadratic Gaussian problem where the objectives are averaged with respect to the noise. The iterative procedure is then called ILQG as presented by Li & Todorov (2007).
Table 1. Convergence properties and oracle costs of Gradient Descent (GD), ILQR, and regularized ILQR (RegILQR) for problem (2) with quadratic $h, g$. The automatic-differentiation cost is stated for problems of the form (26).

<table>
<thead>
<tr>
<th></th>
<th>GD</th>
<th>ILQR</th>
<th>RegILQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global conv. rate</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Local fast conv.</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Oracle cost</td>
<td>$\tau(pd + d^2)$</td>
<td>$\tau p^3d^3$</td>
<td>$\tau p^3d^3$</td>
</tr>
<tr>
<td>Auto-diff cost</td>
<td>$1$</td>
<td>$2d + 1$</td>
<td>$2d + 1$</td>
</tr>
</tbody>
</table>

Proposition 2.1 clarifies that these procedures amount to Gauss-Newton steps which compute

$$\bar{v}^* = \arg\min_{v \in \mathbb{R}^p} q_f (\bar{u} + \bar{v}; \bar{u})$$

(30)

to perform a line-search along the direction $\bar{v}^*$ such that $f(\bar{u} + \alpha \bar{v}^*) \leq f(\bar{u})$. Compared to a Levenberg-Marquardt step (16), that reads

$$\bar{u}^+ = \bar{u} + \arg\min_{v \in \mathbb{R}^p} \left\{ q_f (\bar{u} + \bar{v}; \bar{u}) + \frac{1}{2\gamma_k} \| \bar{v} \|^2 \right\},$$

(31)

we see that this Gauss-Newton step does not take into account the inaccuracy of the model far from the current point. Although a line-search can help ensuring convergence, no rate of convergence is known. For quadratics $h_t, g_t$, the Levenberg-Marquardt steps become regularized Gauss-Newton steps. The regularization term in (31) is critical for convergence to a stationary point.

**Regularized ILQR via regularized Gauss-Newton.** We present convergence guarantees of the regularized Gauss-Newton method for composite optimization problems of the form

$$\min_{\bar{u} \in \mathbb{R}^d} f(\bar{u}) \triangleq h(\bar{x}(\bar{u})) + g(\bar{u}),$$

(32)

where $h: \mathbb{R}^d \to \mathbb{R}$ and $g: \mathbb{R}^p \to \mathbb{R}$ are convex quadratic, and $\bar{x}: \mathbb{R}^p \to \mathbb{R}^d$ is differentiable with continuous gradients. The regularized Gauss-Newton method then naturally leads to a regularized ILQR. In the following, we denote by $L_h$ and $L_g$ the smoothness constants of respectively $h$ and $g$ and by $\ell_{x,S}$ the Lipschitz constant of $\bar{x}$ on the initial sub-level set $S = \{ \bar{u} : f(\bar{u}) \leq f(\bar{u}_0) \}$.

The regularized Gauss-Newton method consists in iterating, starting from a given $\bar{u}_0$,

$$\bar{u}_{k+1} = \bar{u}_k + \arg\min_{v \in \mathbb{R}^p} \left\{ c_f (\bar{u}_k + \bar{v}; \bar{u}_k) + \frac{1}{2\gamma_k} \| \bar{v} \|^2 \right\},$$

(33)

We use $\bar{u}_{k+1}$ to denote (33) hereafter. Convergence is stated in terms of the difference of iterates that, in this case, can directly be linked to the norm of the gradient, denoting $H = \nabla^2 h(\bar{x})$ and $G = \nabla^2 g(\bar{u})$.

$$\bar{u}_{k+1} = \bar{u}_k - (\nabla \bar{x}(\bar{u}_k)H\nabla \bar{x}(\bar{u}_k)^	op + G + \gamma_k^{-1} I_p)^{-1} \nabla f(\bar{u}_k).$$

(34)

Convergence to a stationary point is guaranteed as long as we are able to get a sufficient decrease condition when minimizing this model as stated in the following proposition.

**Proposition 4.1.** Consider composite objectives $f$ as in (32) with convex models $c_f (\cdot; \bar{u})$ defined in (13). Assume that the step sizes $\gamma_k$ of the regularized Gauss-Newton method (33) method are chosen such that

$$f(\bar{u}_{k+1}) \leq f(\bar{u}_k) + \frac{1}{2\gamma_k} \| \bar{u}_{k+1} - \bar{u}_k \|^2$$

(35)

and $\gamma_{\min} \leq \gamma_k \leq \gamma_{\max}$.

Then objective values decrease and the iterates satisfy

$$\min_{k=0,\ldots,N} \| \nabla f(\bar{u}_k) \|^2 \leq \frac{2L(f(\bar{u}_0) - f^*)}{N + 1},$$

where $L = \max_{\gamma \in [\gamma_{\min}, \gamma_{\max}] \gamma (\ell_{x,S} L_h + L_g + \gamma^{-1})^2$ and $f^* = \lim_{k \to +\infty} f(\bar{u}_k)$.

To ensure the sufficient decrease condition, one needs the model to approximate the objective up to a quadratic error which is ensured on any compact set as stated in the following proposition.

**Lemma 4.2.** Consider composite objectives $f$ as in (32) with convex models $c_f (\cdot; \bar{u})$ defined in (13). For any compact set $C \subset \mathbb{R}^p$ there exists $M_C > 0$ such that for any $\bar{u}, \bar{v} \in C$,

$$| f(\bar{v}) - c_f (\bar{v}; \bar{u}) | \leq \frac{M_C}{2} \| \bar{v} - \bar{u} \|^2.$$  (36)

Finally one needs to ensure that the iterates stay in a bounded set which is the case for sufficiently small step-sizes such that the sufficient decrease condition is ensured along the iterates generated by the algorithm.

**Lemma 4.3.** Consider composite objectives $f$ as in (32). For any $k$ such that $\bar{u}_k \in S$, where $S = \{ \bar{u} : f(\bar{u}) \leq f(\bar{u}_0) \}$ is the initial sub-level set, any step-size

$$\gamma_k \leq \hat{\gamma} = \min\{ \ell_{f,S}^{-1}, M_C^{-1} \}$$

(37)

ensures that the sufficient decrease condition (35) is satisfied, where $\ell_{f,S}$ is the Lipschitz constant of $f$ on $S$, $C = S + B_1$ with $B_1$ the unit Euclidean ball and $M_C$ ensures (36).

Combining Proposition 4.1 and Lemma 4.2, we can guarantee that the iterates stay in the initial sub-level set and satisfy the sufficient decrease condition for sufficiently small step-sizes $\gamma_k$. At each iteration the step-size can be found by a line-search guaranteeing sufficient decrease; see (Roulet et al., 2019) for details. The final complexity of the algorithm with line-search then follows.
Algorithm 1: Accelerated Regularized Gauss-Newton

**Input:** Initial point $u_0 \in \mathbb{R}^m$, desired accuracy $\varepsilon$.
**Initialize:** $\alpha_1 := 1$, $\varepsilon_0 := u_0$
**Repeat:** for $k = 1, 2, \ldots$

1: Compute regularized step
   Get $\tilde{v}_k = \text{GN}(\bar{u}_{k-1}; \gamma_k)$ by line-search on $\gamma_k$ s.t.
   $$f(\tilde{v}_k) \leq c_f(\tilde{v}_k; \bar{u}_{k-1}) + (2\gamma_k)^{-1} \|\tilde{v}_k - \bar{u}_{k-1}\|_2^2.$$ 

2: Compute extrapolated step
   - Set $\bar{v}_k = \alpha_k \bar{u}_{k-1} + (1 - \alpha_k) \bar{u}_{k-1}$.
   - Get $\bar{w}_k = \text{GN}(\bar{v}_k; \delta_k)$ by line-search on $\delta_k$ s.t.
     $$f(\bar{w}_k) \leq c_f(\bar{w}_k; \bar{u}_k) + (2\delta_k)^{-1} \|\bar{w}_k - \bar{y}_k\|_2^2. \quad (38)$$
   - Set $\bar{z}_k = \bar{u}_{k-1} + (\bar{w}_k - \bar{u}_{k-1})/\alpha_k$.
   - Pick $\alpha_{k+1} \in (0, 1)$ s.t. $(1 - \alpha_{k+1})/\alpha_k^2 = 1/\alpha_k^2$.
3: Pick best of two steps
   Choose $\bar{u}_k$ such that
   $$f(\bar{u}_k) \leq \min \{f(\tilde{v}_k), f(\bar{w}_k)\} \quad (39)$$
   until $\varepsilon$-near stationarity $\|\nabla f(\bar{u}_k)\| < \varepsilon$

**Corollary 4.4.** For composite objectives $f$ as in (32), the regularized Gauss-Newton method (33) with a decreasing line-search starting from $\gamma_0 \geq \hat{\gamma}$ with decreasing factor $\rho$ finds an $\varepsilon$-stationary point after at most
   $$\frac{2L(f(\bar{u}_0) - f^*)}{\varepsilon^2} + \log(\gamma_0/\hat{\gamma}) / \log(\rho^{-1})$$
calls to the regularized Gauss-Newton oracle, with $\hat{\gamma}$ defined in (37), $f^* = \lim_{k \to +\infty} f(\bar{u}_k)$ and
   $$L = \max_{\gamma \in [\gamma_0, \gamma_0]} \gamma \left( \frac{\rho_2^2 L_h + L_g + \gamma^{-1}}{2} \right)^2.$$ 

**Accelerated regularized Gauss-Newton.** In Algorithm 1 we present an accelerated variant of the regularized Gauss-Newton algorithm that blends a regularized Gauss-Newton step and an extrapolated step to potentially capture convexity in the objective. See (Roulet et al., 2019) for the proof.

**Proposition 4.5.** Consider Algorithm 1 applied to composite objectives $f$ as in (32) with decreasing step-sizes $(\gamma_k)_{k \geq 0}$ and $(\delta_k)_{k \geq 0}$. Then Algorithm 1 satisfies the convergence of the regularized Gauss-Newton method (33) with line-search as presented in Corollary 4.4. Moreover, if the convex models $c_f(\bar{v}; \bar{u})$ defined in (13) lower bound the objective as
   $$c_f(\bar{v}; \bar{u}) \leq f(\bar{v}) \quad (40)$$
for any $\bar{v}, \bar{u} \in \mathbb{R}^p$, then after $N$ iterations of Algorithm 1,
   $$f(\bar{u}_N) - f^* \leq \frac{4\delta^{-1} \|\bar{u}^* - \bar{u}_0\|^2}{(N + 1)^2},$$
where $\delta = \min_{k \in \{1, \ldots, N\}} \delta_k$, $f^* = \min_{\bar{u}} f(\bar{u})$ and $\bar{u}^* \in \text{arg min}_{\bar{u}} f(\bar{u})$.

### 5. Experiments

We illustrate the performance of the algorithms considering including the proposed accelerated regularized Gauss-Newton algorithm on two classical problems drawn from Li & Todorov (2004): swing-up a pendulum, and move a two-links robot arm. Detailed derivations of the control settings are given in (Roulet et al., 2019).

We use the automatic differentiation capabilities of PyTorch (Paszke et al., 2017) to implement the automatic differentiation oracles introduced in Sec. 3. The Gauss-Newton-type steps in Algorithm 1 were computed by solving the dual problem associated as presented in Sec. 3. See Table 1 and (Roulet et al., 2019) for details.

In Figure 1, we compare the convergence, in terms of function value and gradient norm, of ILQR (based on Gauss-Newton), regularized ILQR (based on regularized Gauss-Newton), and accelerated regularized ILQR (based on accelerated regularized Gauss-Newton). These algorithms were presented and introduced in Sec. 4.

For ILQR, we use an Armijo line-search to compute the next step. For both the regularized ILQR and the accelerated regularized ILQR, we use a constant step-size sequence tuned after a burn-in phase of 5 iterations. We leave sophisticated line-search strategies for future work.

The plots show stable convergence of the regularized ILQR on these problems. The proposed accelerated regularized Gauss-Newton algorithm displays stable and fast convergence. Applications of accelerated regularized Gauss-Newton algorithms to reinforcement learning problems would be interesting to explore (Recht, 2018; Fazel et al., 2018; Dean et al., 2018).
Acknowledgements

This work was funded by NIH R01 (#R01EB019335), NSF CCF (#1740551), CPS (#1544797), DMS (#1651851), DMS (#1839371), NRI (#1637748), ONR, RCTA, Amazon, Google, and Honda.

References


