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 through the lens 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.

[1]Department of Statistics, University of Washington, Seattle, USA [2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA [3]Department of Mathematics, University of Washington, Seattle, USA. Correspondence to: Vincent Roulet <vroulet@uw.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

We highlight the equivalence of dynamic programming and gradient back-propagation in this framework and underline the central role of the corresponding automatic differentiation oracle in the complexity analysis in terms of convergence to a stationary point of the objective. We show that the number of calls to this automatic differentiation oracle is the relevant complexity measure given the outreach of machine learning software libraries such as PyTorch or TensorFlow (Paszke et al., 2017; Abadi et al., 2015).

Along the way we propose several improvements to the iterative linear quadratic regulator (ILQR) algorithm, resulting in an accelerated regularized Gauss-Newton algorithm enjoying a complexity bound in terms of convergence to a stationary point and displaying stable convergence behavior in practice. Regularized Gauss-Newton algorithms give a template for the design of algorithms based on partial linearization with guaranteed convergence (Bjorck, 1996; Burke, 1985; Nesterov, 2007; Lewis & Wright, 2016; Drusvyatskiy & Paquette, 2018). The proposed accelerated regularized Gauss-Newton algorithm is based on a Gauss-Newton linearization step stabilized by a proximal regularization and boosted by a Catalyst extrapolation scheme, potentially accelerating convergence while preserving the worst-case guarantee.

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 nonlinear 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 objectives arising in inverse problems (Bjorck, 1996; Nocedal & Wright, 2006; Kaltenbacher et al., 2008; Hansen et al., 2013). The complexity-based viewpoint used in (Nesterov, 2007; Cartis et al., 2011; Drusvyatskiy & Paquette, 2018) informs our analysis and offers generalizations to locally Lipschitz objectives. We build upon these results in particular when equipping the proposed regularized Gauss-Newton algorithm with an extrapolation scheme in the spirit of (Paquette et al., 2018).

All proofs and notations are presented in the longer version of the paper (Roulet et al., 2019). The code for this project is available at `https://github.com/vroulet/ilqc`.

1. Discrete time control

We first present the framework of finite horizon discrete time nonlinear control.

Exact dynamics. Given state variables $x \in \mathbb{R}^d$ and control variables $u \in \mathbb{R}^p$, we consider the control of finite trajectories $\bar{x} = (x_1; \dots; x_\tau) \in \mathbb{R}^{\tau d}$ whose dynamics are controlled by a command $\bar{u} = (u_0; \dots; u_{\tau-1}) \in \mathbb{R}^{\tau p}$, through

$$x_{t+1} = \phi_t(x_t, u_t), \qquad \text{for } t = 0, \dots, \tau - 1, \tag{1}$$

starting from a given $\hat{x}_0 \in \mathbb{R}^d$ until a horizon τ, where the functions $\phi_t : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ 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 nonlinear control problem

$$\min_{\substack{x_0,\dots,x_\tau \in \mathbb{R}^d \\ u_0,\dots,u_{\tau-1} \in \mathbb{R}^p}} \sum_{t=1}^{\tau} h_t(x_t) + \sum_{t=0}^{\tau-1} g_t(u_t) \tag{2}$$
$$\text{subject to} \quad x_{t+1} = \phi_t(x_t, u_t), \qquad x_0 = \hat{x}_0,$$

where, here and thereafter, the dynamics must be satisfied for $t = 0, \dots, \tau - 1$.

Noisy dynamics. The discrepancy between the model and the dynamics can be taken into account by considering noisy dynamics as

$$x_{t+1} = \phi_t(x_t, u_t, w_t), \tag{3}$$

where $w_t \sim \mathcal{N}(0, \mathrm{I}_q)$ for $t = 0, \dots, \tau - 1$. The resulting discrete time control problem consists of optimizing the average cost under the noise $\bar{w} = (w_0; \dots; w_{\tau-1})$ as

$$\min_{\substack{x_0,\dots,x_\tau \in \mathbb{R}^d \\ u_0,\dots,u_{\tau-1} \in \mathbb{R}^p}} \mathbb{E}_{\bar{w}} \left[\sum_{t=1}^{\tau} h_t(x_t) \right] + \sum_{t=0}^{\tau-1} g_t(u_t) \tag{4}$$
$$\text{subject to} \quad x_{t+1} = \phi_t(x_t, u_t, w_t), \qquad x_0 = \hat{x}_0.$$

Costs and penalties. The costs on the trajectory can be used to force the states to follow a given orbit $\hat{x}_1, \dots, \hat{x}_\tau$ as

$$h_t(x_t) = \frac{1}{2}(x_t - \hat{x}_t)^\top Q_t (x_t - \hat{x}_t), \quad \text{with} \quad Q_t \succeq 0, \tag{5}$$

which gives a quadratic tracking problem, while the regularization penalties on the control variables are typically quadratic functions

$$g_t(u_t) = \frac{1}{2} u_t^\top R_t u_t, \quad \text{with} \quad R_t \succ 0. \tag{6}$$

The regularization penalties can also encode constraints on the control variable such as the indicator function of a box

$$g_t(u_t) = \iota_{\{u : c_t^- \le u \le c_t^+\}}(u_t), \quad \text{with} \quad c_t^-, c_t^+ \in \mathbb{R}^p, \tag{7}$$

where ι_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

$$\min_{\substack{y_0,\dots y_\tau \in \mathbb{R}^d \\ v_0,\dots,v_{\tau-1} \in \mathbb{R}^p}} \sum_{t=1}^{\tau} q_{h_t}(x_t^{(k)} + y_t) + \sum_{t=0}^{\tau-1} q_{g_t}(u_t^{(k)} + v_t)$$
$$\text{subject to} \quad y_{t+1} = \ell_{\phi_t}(y_t, v_t), \qquad y_0 = 0, \tag{8}$$

where $\bar{u}^{(k)}$ is the current command, $\bar{x}^{(k)}$ is the corresponding trajectory given by (1), q_{h_t}, q_{g_t} are quadratic approximations of the costs h_t, g_t around respectively $x_t^{(k)}, u_t^{(k)}$ and ℓ_{ϕ_t} is the linearization of ϕ_t around $(x_t^{(k)}, u_t^{(k)})$. The next iterate is then given by $\bar{u}^{(k+1)} = \bar{u}^{(k)} + \alpha \bar{v}^*$ where $\bar{v}^*$ is the solution of (8) and α 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 $\bar{u} \in \mathbb{R}^{\tau p}$, denote by $\tilde{x}(\bar{u}) = (\tilde{x}_1(\bar{u}); \dots; \tilde{x}_\tau(\bar{u})) \in \mathbb{R}^{\tau d}$ the trajectory given by the exact dynamics

$$\tilde{x}_1(\bar{u}) = \phi_0(\hat{x}_0, u_0), \quad \tilde{x}_{t+1}(\bar{u}) = \phi_t(\tilde{x}_t(\bar{u}), u_t). \tag{9}$$

Similarly denote by $\tilde{x}(\bar{u}, \bar{w}) \in \mathbb{R}^{\tau d}$ the trajectory in the noisy case. Denoting the total cost by $h(\bar{x}) = \sum_{t=1}^{\tau} h_t(x_t)$, the total penalty by $g(\bar{u}) = \sum_{t=0}^{\tau-1} g_t(u_t)$, the discrete time control problem (2) with exact dynamics reads

$$\min_{\bar{u} \in \mathbb{R}^{\tau p}} \quad f(\bar{u}) \triangleq h(\tilde{x}(\bar{u})) + g(\bar{u}), \tag{10}$$

and with noisy dynamics,

$$\min_{\bar{u} \in \mathbb{R}^{\tau p}} \quad f(\bar{u}) \triangleq \mathbb{E}_{\bar{w}} \left[h(\tilde{x}(\bar{u}, \bar{w})) \right] + g(\bar{u}), \tag{11}$$

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 ϕ_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 γ corresponds to linearizing both h and $\tilde{x}$ and defining the linear model

$$\ell_f(\bar{u} + \bar{v}; \bar{u}) = \ell_h \left(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v}; \tilde{x}(\bar{u}) \right) + \ell_g(\bar{u} + \bar{v}; \bar{u})$$

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

$$\bar{u}^+ = \bar{u} + \operatorname*{arg\,min}_{\bar{v} \in \mathbb{R}^{\tau p}} \left\{ \ell_f(\bar{u} + \bar{v}; \bar{u}) + \frac{1}{2\gamma} \|\bar{v}\|_2^2 \right\}. \tag{12}$$

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 $\bar{u}$

$$c_f(\bar{u} + \bar{v}; \bar{u}) = h \left(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v} \right) + g(\bar{u} + \bar{v}). \tag{13}$$

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

$$\bar{u}^+ = \bar{u} + \operatorname*{arg\,min}_{\bar{v} \in \mathbb{R}^{\tau p}} \left\{ c_f(\bar{u} + \bar{v}; \bar{u}) + \frac{1}{2\gamma} \|\bar{v}\|_2^2 \right\}. \tag{14}$$

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

$$q_f(\bar{u} + \bar{v}; \bar{u}) = q_h \left(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v}; \tilde{x}(\bar{u}) \right) + q_g(\bar{u} + \bar{v}; \bar{u}), \tag{15}$$

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

$$\bar{u}^+ = \bar{u} + \operatorname*{arg\,min}_{\bar{v} \in \mathbb{R}^{\tau p}} \left\{ q_f(\bar{u} + \bar{v}; \bar{u}) + \frac{1}{2\gamma} \|\bar{v}\|_2^2 \right\}. \tag{16}$$

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 $\bar{u}$

$$m_f(\bar{u} + \bar{v}; \bar{u}) = m_h \left(\tilde{x}(\bar{u}) + \nabla \tilde{x}(\bar{u})^\top \bar{v}; \tilde{x}(\bar{u}) \right) + m_g(\bar{u} + \bar{v}; \bar{u}),$$

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 γ,

$$\bar{u}^+ = \bar{u} + \operatorname*{arg\,min}_{\bar{v} \in \mathbb{R}^{\tau p}} \left\{ m_f(\bar{u} + \bar{v}; \bar{u}) + \frac{1}{2\gamma} \|\bar{v}\|_2^2 \right\}, \tag{17}$$

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* $\bar{u}^+ = \bar{u} + \bar{v}^*$ *where* $\bar{v}^* = (v_0^*; \dots, v_{\tau-1}^*)$ *is the solution of*

$$\min_{\substack{y_0,\dots y_\tau \in \mathbb{R}^d \\ v_0,\dots,v_{\tau-1}\in\mathbb{R}^p}} \sum_{t=1}^{\tau} m_{h_t}(x_t+y_t; x_t) + \sum_{t=0}^{\tau-1} \tilde{m}_{g_t}(u_t+v_t; u_t)$$
$$\text{subject to} \quad y_{t+1} = \Phi_{t,x}^\top y_t + \Phi_{t,u}^\top v_t, \qquad y_0 = 0, \quad (18)$$

where $\tilde{m}_{g_t}(u_t+v_t; u_t) = m_{g_t}(u_t+v_t; u_t) + (2\gamma)^{-1}\|v_t\|_2^2$, $\Phi_{t,x}=\nabla_x \phi_t(x_t, u_t)$, $\Phi_{t,u}=\nabla_u \phi_t(x_t, u_t)$ *and* $x_t = \tilde{x}_t(\bar{u})$.

Proof. Recall that the trajectory defined by $\bar{u}$ reads

$$\tilde{x}_1(\bar{u}) = \phi_0(\hat{x}_0, F_0^\top \bar{u}), \quad \tilde{x}_{t+1}(\bar{u}) = \phi_t(\tilde{x}_t(\bar{u}), F_t^\top \bar{u}),$$

where $F_t = e_{t+1} \otimes \mathrm{I}_p \in \mathbb{R}^{\tau p \times p}$, $e_t \in \mathbb{R}^\tau$ is the t^{th} canonical vector in $\mathbb{R}^\tau$, such that $F_t^\top \bar{u} = u_t$. The gradient reads $\nabla \tilde{x}_1(\bar{u}) = F_0 \nabla_u \phi_0(x_0, u_0)$ followed by

$$\nabla \tilde{x}_{t+1}(\bar{u}) = \nabla \tilde{x}_t(\bar{u}) \nabla_x \phi_t(x_t, u_t) + F_t \nabla_u \phi_t(x_t, u_t),$$

where $x_t = \tilde{x}_t(\bar{u})$ and $x_0 = \hat{x}_0$. For a given $\bar{v} = (v_0; \dots; v_{\tau-1})$, the product $\bar{y} = (y_1; \dots; y_\tau) = \nabla \tilde{x}(\bar{u})^\top \bar{v}$ reads $y_1 = \nabla_u \phi_0(x_0, u_0)^\top v_0$ 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 $y_t = \nabla \tilde{x}_t(\bar{u})^\top \bar{v}$. 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 τ, as presented in the following proposition. The cost is $\mathcal{O}(\tau p^3 d^3)$. 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 τ, 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* τ *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

$$\min_{\substack{y_1,\dots,y_\tau \\ v_0,\dots,v_{\tau-1}}} \sum_{t=1}^{\tau} q_{h_t}(y_t) + \sum_{t=0}^{\tau-1} q_{g_t}(v_t) \qquad (19)$$
$$\text{subject to} \quad y_{t+1} = \ell_t(y_t, v_t), \qquad y_0 = 0,$$

where ℓ_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 $0 \le t \le \tau$, given $\hat{y}_t$, define the cost-to-go from $\hat{y}_t$, as the solution of

$$c_t(\hat{y}_t) = \min_{\substack{y_t,\dots,y_\tau \\ v_t,\dots,v_\tau}} \sum_{t'=t}^{\tau} q_{h_{t'}}(y_{t'}) + \sum_{t'=t}^{\tau-1} q_{g_{t'}}(v_{t'}) \qquad (20)$$
$$\text{subject to} \quad y_{t'+1} = \ell_{t'}(y_{t'}, v_{t'}), \qquad y_t = \hat{y}_t.$$

The cost-to-go functions can be computed recursively by the Bellman equation for $t \in \{\tau-1, \dots, 0\}$,

$$c_t(\hat{y}_t) = q_{h_t}(\hat{y}_t) + \min_{v_t} \{q_{g_t}(v_t) + c_{t+1}(\ell_t(\hat{y}_t, v_t))\} \quad (21)$$

solved for $v_t^*(\hat{y}_t) = \arg\min_{v_t} \{q_{g_{t'}}(v_t) + c_{t+1}(\ell_t(\hat{y}_t, v_t))\}$. The final cost initializing the recursion is defined as $c_\tau(\hat{y}_\tau) = q_{h_\tau}(\hat{y}_\tau)$. 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

$$v_t^* = v_t^*(y_t), \qquad y_{t+1} = \ell_t(y_t, v_t^*). \qquad (22)$$

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 τ. 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 $\mathcal{O}(\tau(pd + d^2))$. 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* $t = 0, \dots, \tau$ *along the trajectory* $x_{t+1} = \phi_t(x_t, u_t)$,

2. *a* backward *pass that computes linear cost-to-go functions as* $c_t(y_t) = \lambda_t^\top y_t + \mu_t$ *where* $\lambda_\tau = \nabla h_\tau(x_\tau)$, $\lambda_t = \nabla h(x_t) + \nabla_x \phi_t(x_t, u_t)\lambda_{t+1}$, *for* $t = \tau - 1, \dots 0$,

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* $t = 0, \dots \tau - 1$.

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 $\ell_{\tilde{x}}(\bar{u} + \bar{v}; \bar{u}, \bar{w}) = \tilde{x}(\bar{u}, \bar{w}) + \nabla_{\bar{u}} \tilde{x}(\bar{u}, \bar{w})^\top \bar{v}$ around the exact linearized trajectory given for $\bar{w} = 0$ reads

$$\hat{\ell}_{\tilde{x}}(\bar{u} + \bar{v}; \bar{u}, \bar{w}) = \tilde{x}(\bar{u}, 0) + \nabla_{\bar{u}} \tilde{x}(\bar{u}, 0)^\top \bar{v} + \nabla_{\bar{w}} \tilde{x}(\bar{u}, 0)^\top \bar{w} + \nabla^2_{\bar{u}\bar{w}} \tilde{x}(\bar{u}, 0)[\bar{u}, \bar{w}, \cdot],$$

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

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

$$\mathbb{E}_{\bar{w}} \left[q_h \left(\hat{\ell}_{\tilde{x}}(\bar{u} + \bar{v}; \bar{u}, \bar{w}); \tilde{x}(\bar{u}, 0) \right) \right]. \tag{23}$$

For simple dynamics ϕ_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* $\nabla^2_{xx} \phi_t$, $\nabla^2_{xw} \phi_t$ *and* $\nabla^2_{ux} \phi_t$ *to be zero. The model minimization step* (17) *for model* (23) *is given by* $\bar{u}^+ = \bar{u} + \bar{v}^*$ *where* $\bar{v}^*$ *is the solution of*

$$\begin{aligned} \min_{\bar{y}, \bar{v}} \quad & \sum_{t=1}^{\tau} \mathbb{E}_{\bar{w}} \left[q_{h_t}(x_t + y_t; x_t) \right] + \sum_{t=0}^{\tau-1} \tilde{q}_{g_t}(u_t + v_t; u_t) \\ \text{s.t.} \quad & y_{t+1} = \Phi_{t,x}^\top y_t + \Phi_{t,u}^\top v_t + \Phi_{t,w}^\top w_t + \phi_{t,w,u}[w_t, u_t, \cdot], \\ & y_0 = 0, \end{aligned} \tag{24}$$

where $\Phi_{t,x} = \nabla_x \phi_t(x_t, u_t, 0)$, $\Phi_{t,u} = \nabla_u \phi_t(x_t, u_t, 0)$, $\Phi_{t,w} = \nabla_w \phi_t(x_t, u_t, 0)$, $\phi_{t,w,u} = \nabla^2_{wu} \phi_t(x_t, u_t, 0)$, $x_t = \tilde{x}_t(\bar{u}, 0)$, *and* $\tilde{q}_{g_t} = q_{g_t} + (2\gamma)^{-1} \|u_t - \cdot\|^2$.

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 analytically for white noise w_t,

$$c_t(\hat{y}_t) = q_{t,x}(\hat{y}_t) + \min_{v_t} q_{t,u}(v_t) + \mathbb{E}_{w_t} \left[c_{t+1}(\ell_t(\hat{y}_t, v_t, w_t)) \right].$$

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

$$\begin{aligned} \min_{\substack{y_0, \dots, y_\tau \in \mathbb{R}^d \\ v_0, \dots, v_{\tau-1} \in \mathbb{R}^p}} \quad & \sum_{t=1}^{\tau} q_{h_t}(y_t) + \sum_{t=0}^{\tau-1} q_{g_t}(v_t) \\ \text{subject to} \quad & y_{t+1} = \ell_t(y_t, v_t), \quad y_0 = 0, \quad v_t \in \mathcal{U}_t, \end{aligned} \tag{25}$$

where $\mathcal{U}_t = \{u : C_t u \leq d_t\}$, q_{h_t} are convex quadratics, q_{g_t} are strictly convex quadratics and ℓ_t are linear dynamics. Interior point methods introduce a log-barrier function $\mathcal{B}_t(u) = \log(d_t - C_t u)$ and minimize

$$\begin{aligned} \min_{\substack{y_0, \dots, y_\tau \in \mathbb{R}^d \\ v_0, \dots, v_{\tau-1} \in \mathbb{R}^p}} \quad & \sum_{t=1}^{\tau} q_{h_t}(y_t) + \sum_{t=0}^{\tau-1} q_{g_t}(v_t) + \mu_k \mathcal{B}_t(v_t) \\ \text{subject to} \quad & y_{t+1} = \ell_t(y_t, v_t), \qquad y_0 = 0, \end{aligned}$$

where μ_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

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

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

$$\min_{\bar{u} \in \mathbb{R}^{\tau p}} h(\xi(\bar{u})) + g(\bar{u}), \tag{26}$$

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

$$\tilde{x}_1(\bar{u}) = \phi_0(\hat{x}_0, u_0), \quad \tilde{x}_{t+1}(\bar{u}) = \phi_t(\tilde{x}_t(\bar{u}), u_t), \tag{27}$$

with $\hat{x}_0 \in \mathbb{R}^d$ and continuously differentiable functions $\phi_t : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}$.

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 $\nabla\xi(\bar{u})z$ for $z \in \mathbb{R}^d$. In our setting, this amounts to solving $\min_{\bar{v}\in\mathbb{R}^{\tau p}} -z^\top \nabla\xi(\bar{u})^\top \bar{v} + \frac{1}{2}\|\bar{v}\|_2^2$ 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 $\xi : \mathbb{R}^{\tau p} \to \mathbb{R}^d$ defined by successive compositions as in (27), $\bar{u} \in \mathbb{R}^{\tau p}$ 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 $\xi : \mathbb{R}^{\tau p} \to \mathbb{R}^d$ defined by successive compositions as in (27), an automatic-differentiation oracle and a differentiable function $f : \mathbb{R}^{\tau p} \to \mathbb{R}$, the derivative of $z \to f(\nabla\xi(\bar{u})z)$ for $\bar{u} \in \mathbb{R}^{\tau p}$ 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

$$\min_{\bar{v}\in\mathbb{R}^{\tau p}} h\left(\xi(\bar{u})+\nabla\xi(\bar{u})^\top\bar{v}\right)+g(\bar{u}+\bar{v})+\frac{1}{2\gamma}\|\bar{v}\|_2^2. \quad (28)$$

For smooth objectives h and g and Lipschitz continuous dynamics ϕ_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 ϕ_t, a regularized Gauss-Newton step given by (28) is solved up to ε accuracy by a fast gradient method with at most*

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

calls to an automatic-differentiation oracle, where M_ξ is the Lipschitz continuity constant of ξ, 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

$$\min_{\bar{v}\in\mathbb{R}^{\tau p}} q_h\left(\xi(\bar{u})+\nabla\xi(\bar{u})^\top\bar{v};\xi(\bar{u})\right)+q_g(\bar{u}+\bar{v};\bar{u})+\frac{1}{2\gamma}\|\bar{v}\|_2^2, \quad (29)$$

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 $\bar{x} = \tilde{x}(\bar{u})$, DDP consists in approximating the cost-to-go functions as

$$\begin{aligned} c_t(y) =& q_{h_t}(x_t+y;x_t) + \min_v\{q_{g_t}(u_t+v;u_t) \\ &+ q_{c_{t+1}\circ\phi_t}(x_t+y,u_t+v;x_t,u_t)\}, \end{aligned}$$

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).

	GD	ILQR	RegILQR
Global conv. rate	Yes	No	Yes
Local fast conv.	No	Yes	Yes
Oracle cost	$\tau(pd+d^2)$	$\tau p^3 d^3$	$\tau p^3 d^3$
Auto-diff cost	1	$2d+1$	$2d+1$

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) .

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

$$\bar{v}^* = \operatorname*{arg\,min}_{\bar{v}\in\mathbb{R}^{\tau p}} q_f(\bar{u}+\bar{v};\bar{u}) \tag{30}$$

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

$$\bar{u}^+ = \bar{u} + \operatorname*{arg\,min}_{\bar{v}\in\mathbb{R}^{\tau p}} \left\{ q_f(\bar{u}+\bar{v};\bar{u}) + \frac{1}{2\gamma}\|\bar{v}\|_2^2 \right\}, \tag{31}$$

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

$$\min_{\bar{u}\in\mathbb{R}^{\tau d}} f(\bar{u}) \triangleq h(\tilde{x}(\bar{u})) + g(\bar{u}), \tag{32}$$

where $h : \mathbb{R}^{\tau d} \to \mathbb{R}$ and $g : \mathbb{R}^{\tau p} \to \mathbb{R}$ are convex quadratic, $\tilde{x} : \mathbb{R}^{\tau p} \to \mathbb{R}^{\tau d}$ 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 $\ell_{\tilde{x},S}$ the Lipschitz constant of $\tilde{x}$ 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 $\bar{u}_0$,

$$\bar{u}_{k+1} = \bar{u}_k + \operatorname*{arg\,min}_{\bar{v}\in\mathbb{R}^{\tau p}} \left\{ c_f(\bar{u}_k+\bar{v};\bar{u}_k) + \frac{1}{2\gamma_k}\|\bar{v}\|_2^2 \right\}, \tag{33}$$

We use $\bar{u}_{k+1} = \mathrm{GN}(u_k;\gamma_k)$ 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 $G = \nabla^2 g(\bar{u})$,

$$\bar{u}_{k+1} = \bar{u}_k - (\nabla\tilde{x}(\bar{u}_k) H \nabla\tilde{x}(\bar{u}_k)^\top + G + \gamma_k^{-1}\,\mathrm{I}_{\tau p})^{-1}\nabla f(\bar{u}_k). \tag{34}$$

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 γ_k of the regularized Gauss-Newton method (33) method are chosen such that*

$$f(\bar{u}_{k+1}) \leq c_f(\bar{u}_{k+1};\bar{u}_k) + \frac{1}{2\gamma_k}\|\bar{u}_{k+1}-\bar{u}_k\|_2^2 \tag{35}$$

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

Then objective values decrease and the iterates satisfy

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

where $L = \max_{\gamma\in[\gamma_{\min},\gamma_{\max}]} \gamma(\ell_{\tilde{x},S}^2 L_h + L_g + \gamma^{-1})^2$ 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 $C \subset \mathbb{R}^{\tau p}$ there exists $M_C > 0$ such that for any $\bar{u}, \bar{v} \in C$,*

$$|f(\bar{v}) - c_f(\bar{v};\bar{u})| \leq \frac{M_C}{2}\|\bar{v}-\bar{u}\|_2^2. \tag{36}$$

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*

$$\gamma_k \leq \hat{\gamma} = \min\{\ell_{f,S}^{-1}, M_C^{-1}\} \tag{37}$$

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 γ_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 $\bar{u}_0 \in \mathbb{R}^m$, desired accuracy ε.
Initialize: $\alpha_1 := 1$, $\bar{z}_0 := \bar{u}_0$
Repeat: for $k = 1, 2, \ldots$

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

2: Compute extrapolated step
- Set $\bar{y}_k = \alpha_k \bar{z}_{k-1} + (1-\alpha_k)\bar{u}_{k-1}$.
- Get $\bar{w}_k = \text{GN}(\bar{y}_k; \delta_k)$ by line-search on δ_k s.t.
$f(\bar{w}_k) \leq c_f(\bar{w}_k; \bar{y}_k) + (2\delta_k)^{-1}\|\bar{w}_k - \bar{y}_k\|_2^2$. (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. $(1-\alpha_{k+1})/\alpha_{k+1}^2 = 1/\alpha_k^2$.

3: Pick best of two steps
Choose $\bar{u}_k$ such that
$$f(\bar{u}_k) \leq \min\{f(\bar{v}_k), f(\bar{w}_k)\} \tag{39}$$

until ε-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 ρ finds an ε-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

$$L = \max_{\gamma \in [\hat{\gamma}, \gamma_0]} \gamma(\ell_{\tilde{x},S}^2 L_h + L_g + \gamma^{-1})^2.$$

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*

$$c_f(\bar{v}; \bar{u}) \leq f(\bar{v}) \tag{40}$$

for any $\bar{u}, \bar{v} \in \mathbb{R}^{\tau p}$, 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 $\bar{u}^* \in \arg\min_{\bar{u}} f(\bar{u})$.

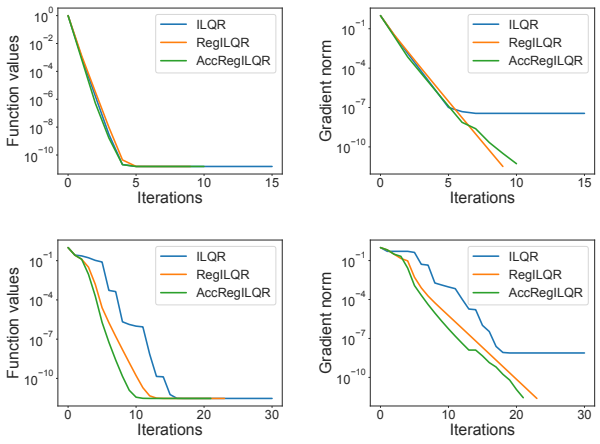

Figure 1. Convergence of ILQR, regularized ILQR and accelerated regularized ILQR on the inverted pendulum (top) and two-links arm (bottom) control problems for an horizon $\tau = 100$.

5. Experiments

We illustrate the performance of the algorithms considered 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

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/.

Bellman, R. *Introduction to the mathematical theory of control processes*, volume 2. Academic press, 1971.

Bertsekas, D. P. *Dynamic programming and optimal control*. Athena Scientific, 3rd edition, 2005.

Bjorck, A. *Numerical methods for least squares problems*. SIAM, 1996.

Burke, J. V. Descent methods for composite nondifferentiable optimization problems. *Mathematical Programming*, 33(3):260–279, 1985.

Cartis, C., Gould, N. I. M., and Toint, P. L. On the evaluation complexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM Journal on Optimization*, 21(4):1721–1739, 2011.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. Regret bounds for robust adaptive control of the linear quadratic regulator. In *Advances in Neural Information Processing Systems*, pp. 4188–4197, 2018.

Dontchev, A. L., Huang, M., Kolmanovsky, I. V., and Nicotra, M. M. Inexact Newton-Kantorovich methods for constrained nonlinear model predictive control. *IEEE Transactions on Automatic Control*, 2018.

Drusvyatskiy, D. and Paquette, C. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, pp. 1–56, 2018.

Dunn, J. C. and Bertsekas, D. P. Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems. *Journal of Optimization Theory and Applications*, 63(1):23–38, 1989.

Fazel, M., Ge, R., Kakade, S., and Mesbahi, M. Global convergence of policy gradient methods for the linear quadratic regulator. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 2018.

Griewank, A. and Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Grüne, L. and Pannek, J. *Nonlinear model predictive control*. Springer, 2017.

Hansen, P. C., Pereyra, V., and Scherer, G. *Least squares data fitting with applications*. JHU Press, 2013.

Jacobson, D. H. and Mayne, D. Q. *Differential Dynamic Programming*. Elsevier, 1970.

Kakade, S. M. and Lee, J. D. Provably correct automatic sub-differentiation for qualified programs. In *Advances in Neural Information Processing Systems*, pp. 7125–7135, 2018.

Kaltenbacher, B., Neubauer, A., and Scherzer, O. *Iterative regularization methods for nonlinear ill-posed problems*, volume 6. Walter de Gruyter, 2008.

LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pp. 21–28, 1988.

Lewis, A. S. and Wright, S. J. A proximal method for composite minimization. *Mathematical Programming*, 158: 501–546, 2016.

Li, W. and Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *1st International Conference on Informatics in Control, Automation and Robotics*, volume 1, pp. 222–229, 2004.

Li, W. and Todorov, E. Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. *International Journal of Control*, 80(9):1439–1453, 2007.

Liao, L.-Z. and Shoemaker, C. A. Advantages of differential dynamic programming over Newton's method for discrete-time optimal control problems. Technical report, Cornell University, 1992.

Mayne, D. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

Nesterov, Y. Modified Gauss-Newton scheme with worst case guarantees for global performance. *Optimization Methods & Software*, 22(3):469–483, 2007.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, 2nd edition, 2006.

Paquette, C., Lin, H., Drusvyatskiy, D., Mairal, J., and Harchaoui, Z. Catalyst for gradient-based nonconvex optimization. In *21st International Conference on Artificial Intelligence and Statistics*, pp. 1–10, 2018.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch, 2017. URL https://pytorch.org/.

Recht, B. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

Richter, S., Jones, C. N., and Morari, M. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57(6):1391–1403, 2012.

Roulet, V., Srinivasa, S., Drusvyatskiy, D., and Harchaoui, Z. Iterative linearized control: Stable algorithms and complexity guarantees. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Sideris, A. and Bobrow, J. E. An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems. In *Proceedings of the American Control Conference*, pp. 2275–2280, 2005.

Tassa, Y., Mansard, N., and Todorov, E. Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation*, pp. 1168–1175, 2014.

Todorov, E. and Li, W. Optimal control methods suitable for biomechanical systems. In *Proceedings of the 25th Annual International Conference of the IEEE*, volume 2, pp. 1758–1761, 2003.