Sequential Operator Splitting for Constrained Nonlinear Optimal Control

Vikas Sindhwani[1] Rebecca Roelofs[2] Mrinal Kalakrishnan[3]

***Abstract*— We develop TROSS, a solver for constrained non-smooth trajectory optimization based on a sequential operator splitting framework. TROSS iteratively improves trajectories by solving a sequence of subproblems setup within evolving trust regions around current iterates using the Alternating Direction Method of Multipliers (ADMM). TROSS achieves consensus among competing objectives, such as finding low-cost dynamically feasible trajectories respecting control limits and safety constraints. A library of building blocks in the form of inexpensive and parallelizable proximal operators associated with trajectory costs and constraints can be used to configure the solver for a variety of tasks. The method shows faster cost reduction compared to iterative Linear Quadratic Regulator (iLQR) and Sequential Quadratic Programming (SQP) on a control-limited vehicle maneuvering task. We demonstrate TROSS on shortest-path navigation of a variant of Dubin's car in the presence of obstacles, while exploiting passive dynamics of the system. When applied to a constrained robust state estimation problem involving nondifferentiable nonconvex penalties, TROSS shows less susceptibility to non-Gaussian dynamics disturbances and measurement outliers in comparison to an Extended Kalman smoother. Unlike generic SQP methods, our approach produces time-varying linear feedback control policies even for constrained control tasks. The solver is potentially suitable for nonlinear model predictive control and moving horizon state estimation in embedded systems.**

I. INTRODUCTION

Let $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ denote a deterministic discrete-time nonlinear dynamical system with states $\mathbf{x}_t \in \mathbb{R}^n$ and controls $\mathbf{u}_t \in \mathbb{R}^m$. A trajectory τ over a time horizon T is a sequence of state-control pairs $(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1, \ldots \mathbf{x}_{T-1}, \mathbf{u}_{T-1}, \mathbf{x}_T)$. We consider the following class of constrained optimization problems over trajectory variables $\tau \in \mathbb{R}^{mT+n(T+1)}$,

$$\underset{\tau \in \mathcal{C}}{\operatorname{argmin}} \quad c(\tau), \tag{1}$$

$$\text{subject to: } \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad t = 0 \ldots (T-1) \tag{2}$$

$$\tau_{min} \leq \tau \leq \tau_{max} \tag{3}$$

Above, we seek a trajectory that minimizes the cost function, $c(\tau)$, while respecting the dynamics of the system (Eqn 2), and satisfying other operational constraints. The bound constraints in Eqn 3, may be imposed to keep control inputs within physical limits of actuators and to guide the evolution of states along specific waypoints in the state space. More complex constraints may be encoded in the set $\mathcal{C}$ in Eqn. 1. For example, in robot motion planning problems, acceptable trajectories also need to avoid collision with obstacles in the environment, and feasible states might live on a nonlinear manifold (e.g., pose variables in $SE(3)$). The cost function c is not necessarily convex, differentiable or separable across time; the dynamics is typically highly nonlinear, and the constraint set $\mathcal{C}$ may in general be non-convex. The trajectory optimization problem above may be instantiated for a given fixed initial state $\hat{\mathbf{x}}_0$ in which case we include the constraint $\mathbf{x}_0 = \hat{\mathbf{x}}_0$, or the initial state might be a free optimization variable, as is natural in state estimation settings. In the context of robotics, the delicate interplay between the geometry of the environment on one hand, and nonlinear dynamics and physical limits of underactuated systems on the other, makes optimization problems of the above form highly challenging.

Differential Dynamic Programming (DDP) [13] and Iterative Linear Quadratic Regulator (iLQR) [16] are among the most effective methods for unconstrained optimal control problems involving twice-differentiable time-separable cost functions. They are variations of Newton's method [7] and belong to the family of shooting or indirect methods for trajectory optimization where states are treated as implicit functions of the control sequence, as opposed to explicit optimization variables. At each iteration, a time-varying Linear Quadratic Regulator (TV-LQR) subproblem is solved, by linearizing the dynamics and constructing a quadratic approximation to the cost function, yielding a direction along which the current trajectory is updated via line search. DDP also incorporates second order dynamics approximation in this subproblem, while iLQR ignores it for efficiency. By construction, these methods maintain dynamics feasibility at each iteration and get their efficiency from linear time (in T) Ricatti recursion solution to the TV-LQR problem. The optimal controls are expressed in the form of locally linear feedback control policies that can be used for robust closed loop execution. An extension of iLQR for handling bound constraints on control variables was proposed in [22].

When additional state and control constraints are introduced, the problem is typically treated as an instance of general nonlinear programming. In such "direct methods", state and control variables are jointly optimized in the presence of dynamics satisfaction constraints. Sequential Quadratic Programming (SQP) [23] or Sequential Convex Programming (SCP) techniques, e.g., as implemented in SNOPT [10] or TrajOpt [21] packages, are among the preferred approaches for solving such problems. Constrained problems cast in terms of an SQP lose the computational efficiency of Ricatti recursions (though they can exploit

[1]Google Brain, New York, USA sindhwani@google.com

[2]Dept. of Computer Science, Univ. of California, Berkeley CA, USA. Research done during Google internship. roelofs@cs.berkeley.edu

[3]X, Mountain View CA, USA kalakris@x.team

structure in the problem in the form of Jacobian and Hessian sparsity), and only return an open loop sequence of controls as opposed to feedback policies.

In this paper, we develop a *sequential operator splitting* framework for solving constrained trajectory optimization problems. Our framework, called TROSS (Trajectory Optimization with Sequential Splitting) has the following features:

- it does not require twice-differentiability of cost functions, and can also optimize non-differentiable and non-time-separable cost functions.
- its implementation is organized around a library of simple and parallelizable building blocks of projection and proximal operators.
- unlike SQP techniques but like iLQR, TROSS returns closed-loop locally linear feedback policies through a proximal operator associated with a local LQR problem.
- we demonstrate TROSS on the following problems,
  - A maneuvering task involving a car-like robot with control limits. Here, we compare the performance of TROSS against the control-limited variant of iLQR developed in [22], and also against SQP and other nonlinear programming methods.
  - An obstacle avoidance problem involving navigation of a Dubin-like car [6] demonstrating the ability of TROSS to handle geometric safety constraints on state variables together with sparsity inducing regulation of controls for exploiting passive dynamics [17] of the system.
  - A state estimation [12] problem of reconstructing the trajectory of a target from corrupted range measurements recorded from a radar ground station [14]. Here, we show that TROSS can optimize trajectories with respect to non-smooth non-convex cost functions such as the capped-l_1 loss. On this problem, we include a comparison with Extended Kalman Filtering (EKF).

  TROSS may potentially also be well-suited to real-time on-device applications of nonlinear model predictive control [8], [23] and moving horizon estimation requiring low-to-medium precision. Its use as a flexible supervisor for guided policy search [15] is another potential application

TROSS is a trust-region based approach that solves a sequence of local constrained control subproblems using operator splitting techniques, and in particular, the Alternating Direction Method of Multipliers (ADMM) [2]. The application of ADMM techniques to optimal control problems was proposed in [18] for time-separable convex quadratic objectives and linear dynamics in the presence of additional constraints and regularizers that admit simple proximal/projection operators. Recently, [4] demonstrated that ADMM can also often be remarkably effective for minimizing convex objectives over specific classes of non-convex sets. The TROSS framework may be viewed as a natural extension of these efforts: we do not assume convexity or separability of the cost function, and handle nonlinear dynamics via iterative linearization. The cost function and constraints may be replaced by local approximations that admit an inexpensive proximal or projection operator. The resulting subproblem is solved by ADMM in the presence of a trust region constraint that enforces the region of validity of these approximations. This subproblem need not be convex and does not require very precise solutions. In this splitting framework, cost minimization and constraint satisfaction including dynamics feasibility, are separated allowing a sequence of ADMM solves to attempt to bring consensus between competing control objectives. In the outer loop, the trust region radius is adjusted using the "filter" strategy [9] which maintains a Pareto frontier of cost and constraint satisfaction pairs, to decide whether to expand or contract the trust region.

We start with a quick background on SCP techniques and ADMM, and then introduce the TROSS framework. Numerical results are presented in Section IV.

II. Background

A. Sequential Convex Programming

Sequential Convex Programming (SCP) refers to a heuristic strategy for solving general nonlinear programming problems via a sequence of convex minimization subproblems. Suppose $f : \mathbb{R}^n \mapsto \mathbb{R}$ needs to be minimized subject to equality constraints, $h(\mathbf{x}) = 0$, and inequality constraints $g(\mathbf{x}) \leq 0$, SCP constructs a proxy subproblem at each iteration k, where the objective and constraint functions are approximated by convex functions around the current iterate $\mathbf{x}^k$. The validity of these approximations is assumed to hold within a convex trust region $\mathcal{T}^k$, which is explicitly included among the constraints. The convexification can be done via linearization, or using second order Taylor approximations with suitable Hessian modifications, or even building derivative-free regional convex models using regression methods. If the solution to this subproblem yields progress in terms of some measure of improvement in objective value and constraint satisfaction, then the iterate is accepted as the next step. Otherwise, the trust region is reduced and the problem is resolved.

The practical implementation of SCP solvers requires considerable care. In particular, it involves (1) choosing an appropriate structure-exploiting solver for the convex subproblems and controlling its numerical stability and level of precision, (2) properly adapting the trust region radius along the optimization trajectory, and (3) handling infeasibility that might be encountered if local convexification generates incompatible constraints or if the trust region shrinks so much as to prevent a feasible update. One way to avoid infeasibility is to solve relaxed problems by introducing a set of non-negative slack variables to the convexified objective function [21].

B. Consensus ADMM

Consider optimization problems of form,

$$\underset{\mathbf{x}}{\text{argmin}} \quad \sum_{i=1}^{l} f_i(\mathbf{x}) + g(\mathbf{x}) \tag{4}$$

where each f_i is a convex term in the objective, and g is a secondary term representing a simple convex constraint or regularization. A constraint set $\mathcal{C}$ may be encoded in the above form by setting one of the cost terms to be the associated indicator function denoted by $\mathbb{1}_{\mathcal{C}}$. The problem may be equivalently rewritten in global variable consensus form as follows,

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{argmin}} \quad \sum_{i=1}^{l} f_i(\mathbf{x}_i) + g(\bar{\mathbf{x}}) \tag{5}$$

$$\mathbf{x}_i = \bar{\mathbf{x}}, \quad i = 1 \dots l \tag{6}$$

Here, variables are split into l local copies each myopically responsible for their associated cost reduction or constraint satisfaction, with a requirement to eventually achieve consensus with "global" variables $\bar{\mathbf{x}}$. The scaled form updates of the ADMM [2] solver for this problem are as follows,

$$\mathbf{x}_i^{j+1} = \texttt{prox}_{\rho,f_i}[\bar{\mathbf{x}}^j - \lambda_0^j] \tag{7}$$

$$\bar{\mathbf{x}}^{j+1} = \texttt{prox}_{l\rho,g}\Big[\frac{1}{l}\sum_{i=1}^{l}(\mathbf{x}_i^{j+1} + \lambda_i^j)\Big] \tag{8}$$

$$\lambda_i^{j+1} = \lambda_i^j + \mathbf{x}_i^{j+1} - \bar{\mathbf{x}}^{j+1}, \quad i = 1 \dots l \tag{9}$$

where $\mathbf{x}_i$ and λ_i denote primal and dual variables, j is the iteration index, $\rho > 0$ is the ADMM penalty parameter, and $\texttt{prox}_{\rho,f}$ denotes the proximal operator associated with the function f:

$$\texttt{prox}_{\rho,f}[\hat{\mathbf{x}}] = \underset{\mathbf{x}}{\text{argmin}}\, f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{10}$$

The proximal operator for the indicator function of a constraint set $\mathcal{C}$ reduces to the projection operator,

$$\texttt{proj}_{\mathcal{C}}[\hat{\mathbf{x}}] = \underset{\mathbf{x}\in\mathcal{C}}{\text{argmin}} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{11}$$

At iteration j, the primal and dual residual norms, denoted $\|\mathbf{r}_p\|_2^2, \|\mathbf{r}_d\|_2^2$ respectively, turn out to be interpretable as natural l_2 measures of consensus:

$$\|\mathbf{r}_p\|_2^2 = \sum_{i=1}^{l}\|\mathbf{x}_i^j - \bar{\mathbf{x}}^j\|_2^2, \quad \|\mathbf{r}_d\|_2^2 = l\rho^2\|\bar{\mathbf{x}}^j - \bar{\mathbf{x}}^{j-1}\|_2^2 \tag{12}$$

For a feasible convex optimization problem, consensus ADMM converges to the globally optimal solution. The primal and dual residual norms, which can be used to define appropriate stopping criteria, converge to zero. Under certain conditions, convergence is also guaranteed for various update orders, frequencies and inexactness of projections and proximal operations. Furthermore, [20] show that ADMM handles infeasible bound-constrained Quadratic Programs gracefully: the primal iterates converge to a minimizer of the Euclidean distance between the subspace defined by equality constraints and the convex set defined by bounds. ADMM has also been found to be effective for certain non-convex problems [4], [24], typically those where proximal/projection operators can be computed exactly despite non-convexity.

III. Sequential Operator Splitting

We rewrite the optimal control problem in Eqn. 1-3 compactly as,

$$\underset{\tau}{\text{argmin}}\, c(\tau) + \mathbb{1}_{\mathcal{F}}(\tau) + \mathbb{1}_{\mathcal{B}}(\tau) + \mathbb{1}_{\mathcal{C}}(\tau) \tag{13}$$

where,

- $\mathcal{F} = \{\tau : \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t),\ t = 0 \dots (T-1)\}$ denotes the set of trajectories that are dynamically feasible. The initial state $\mathbf{x}_0$ may be prescribed for this set.
- $\mathcal{B} = \{\tau : \tau_{min} \le \tau \le \tau_{max}\}$ denotes trajectories that respect the specified bound constraints
- $\mathcal{C}$ specifies additional trajectory constraints.

We now describe the `TROSS` solver. The method has an outer SCP-like loop where at step k, we maintain a current solution trajectory τ^k. We use the notation $\mathbf{u}_t^k, \mathbf{x}_t^k$ to denote the control and state components of τ^k. Around, τ^k we construct a control subproblem as follows.

First, in the neighborhood of τ^k, we approximately decompose the trajectory cost as follows,

$$c(\tau) \approx q^k(\tau) + \bar{q}^k(\tau)$$

where q^k is a time-separable quadratic, and $\bar{q}^k$ is a non-separable quadratic or non-quadratic piece of the approximation.

A local dynamics feasibility set $\mathcal{F}^k$ is defined through linearization,

$$\delta\mathbf{x}_{t+1} = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_t^k)\delta\mathbf{x_t} + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{u}_t^k)\delta\mathbf{u}_t + \mathbf{c}_t \tag{14}$$

where $\delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_t^k$, $\delta\mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_t^k$ and $\mathbf{c}_t = f(\mathbf{x}_t^k, \mathbf{u}_t^k) - \mathbf{x}_{t+1}^k$.

The constraint set $\mathcal{C}$ may also be approximated by a set $\mathcal{C}^k$. For example, if $\mathcal{C}$ is defined through state inequality constraints $g(\mathbf{x}_t) \le 0$, then $\mathcal{C}^k$ may be the polyhedral approximation,

$$g(\mathbf{x}_t^k) + \frac{\partial g}{\partial \mathbf{x}}(\mathbf{x}_t^k)\delta\mathbf{x}_t \le 0 \tag{15}$$

Note that components of the cost function and constraint set might be left as is if they already admit a fast and accurate proximal operator.

Finally, each trajectory variable τ_i is associated with a trust region radius μ_i^k (for simplicity, in our implementation, we associate one radius for all control variables and one radius for all state variables). We use box-shaped trust region constraints, $\mathcal{T}^k = \{\tau : |\tau_i - \tau_i^k| \le \mu_i^k\}$ though other trust region geometries (e.g. spheres or ellipsoids) can also be supported. These trust region constraints and the original bound constraints can be clubbed together into the constraint set $\mathcal{B}^k = \mathcal{B} \cap \mathcal{T}^k$ defined by,

$$\max(\tau^k - \mu^k, \tau_{min}) \le \tau \le \min(\tau^k + \mu^k, \tau_{max}) \tag{16}$$

.

The local subproblem can now be defined,

$$\underset{\tau}{\text{argmin}} \quad \big[q^k(\tau) + \mathbb{1}_{\mathcal{F}^k}(\tau)\big] + \bar{q}^k(\tau) + \mathbb{1}_{\mathcal{C}^k}(\tau) + \mathbb{1}_{\mathcal{B}^k}(\tau) \tag{17}$$

This is in the form of Eqn 4 where the first three terms may be identified with f_i's and the last bound constraints term may be identified with the function g. We now apply the ADMM update equations in Eqn. 7-9 to solve the subproblem. Using the notation $q^k_{\mathcal{F}^k} = q^k + \mathbb{1}_{\mathcal{F}^k}$, these updates take the following form for $l = 3$,

$$\tau_1^{j+1} = \texttt{prox}_{\rho, q^k_{\mathcal{F}^k}}[\bar{\tau}^j - \lambda_1^j] \quad (18)$$

$$\tau_2^{j+1} = \texttt{prox}_{\rho, \bar{q}^k}[\bar{\tau}^j - \lambda_2^j] \quad (19)$$

$$\tau_3^{j+1} = \texttt{proj}_{\mathcal{C}^k}[\bar{\tau}^j - \lambda_3^j] \quad (20)$$

$$\bar{\tau}^{j+1} = \texttt{proj}_{\mathcal{B}^k}\Big[\frac{1}{l}\sum_{i=1}^{l}(\tau_i^{j+1} + \lambda_i^j)\Big] \quad (21)$$

$$\lambda_i^{j+1} = \lambda_i^j + \tau_i^{j+1} - \bar{\tau}^{j+1}, \quad i = 1 \dots l \quad (22)$$

If the solution returned by ADMM to this subproblem shows sufficient improvement in cost reduction or constraint satisfaction relative to the previous iterate, we accept the step in the outer loop. We then proceed to construct a new subproblem around the latest solution. The next ADMM is warmstarted by the previous solution. If neither cost nor constraint satisfaction improves significantly, the step is rejected and the trust region radius is contracted. Because one of the primal trajectories is associated with a time-varying LQR problem (Eqn. 18), upon achieving near consensus, the solution is accompanied by feedback gain matrices. As such, DDP-style updates may be performed in the outer loop by propagating states and controls through the original nonlinear dynamics. Note that several variations on this theme are possible with different choices for splitting, variable update orders and choice of trust region geometry.

In subsection A, we describe a Ricatti recursion procedure to rapidly compute the proximal operator for the time-varying linear quadratic control problem in Eqn.18 for changing inputs across ADMM iterations. In subsection B, we collect some proximal operators for non-quadratic and non-separable cost functions, Eqn. 19, for applications demonstrated in this paper in section IV. The projection onto state/control limits and trust region constraints is described in subsection C. In subsection D, we briefly describe approximate projection operators, Eqn. 20, for obstacle avoidance settings. Finally, subsection E briefly describes our trust region adaptation strategy.

A. Proximal Operator for Time-Varying LQR

A second order Taylor approximation to a time-separable trajectory cost function has the general form,

$$q(\tau) = \sum_{t=0}^{T}\left(\frac{1}{2}\mathbf{x}_t^T\mathbf{Q}_t\mathbf{x}_t + \mathbf{q}_t^T\mathbf{x}_t\right) + \quad (23)$$

$$\sum_{t=0}^{T-1}\left(\frac{1}{2}\mathbf{u}_t^T\mathbf{R}_t\mathbf{u}_t + \mathbf{r}_t^T\mathbf{u}_t\right) + \mathbf{u}_t^T\mathbf{M}_t\mathbf{x}_t \quad (24)$$

The domain of this approximation is restricted to trajectories that satisfy linearized dynamics $\mathcal{F} = \{\tau : \mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{c}_t\}$ where the initial state $\mathbf{x}_0$ might be fixed or free. The proximal operator associated with this restricted quadratic (denoted by $q_{\mathcal{F}}$), as required in Eqn. 18, is

$$\texttt{prox}_{q_{\mathcal{F}},\rho}[\hat{\tau}] = \underset{\tau\in\mathcal{F}}{\text{argmin}}\, q(\tau) + \frac{\rho}{2}\|\rho - \hat{\rho}\|_2^2,$$

which is,

$$\underset{\tau\in\mathcal{F}}{\text{argmin}} \sum_{t=0}^{T}\frac{1}{2}\mathbf{x}_t^T(\mathbf{Q}_t + \rho I_n)\mathbf{x}_t + (\mathbf{q}_t^T - \rho\hat{\mathbf{x}}_t)^T\mathbf{x}_t$$

$$+\sum_{t=0}^{T-1}\frac{1}{2}\mathbf{u}_t^T(\mathbf{R}_t + \rho I_m)\mathbf{u}_t + (\mathbf{r}_t^T - \rho\hat{\mathbf{u}}_t^T)^T\mathbf{u}_t + \mathbf{u}_t^T\mathbf{M}_t\mathbf{x}_t \quad (25)$$

where $\mathbf{I}_n$ denotes the $n\times n$ identity matrix. This computation is simply a modified LQR problem, which may be solved using classic Ricatti recursion. The complexity of a single solve is $O(T(m+n)^3)$. In the context of using ADMM updates in Eqn. 18, this proximal operator needs to be repeatedly called for varying $\hat{\tau}$. In this case, the cost can be reduced to $O(T(m+n)^2)$ for each repeated solve after a preprocessing step where certain factorizations can be cached for a $O(T(m+n)^3)$ one-time cost. Note that [18] consider the same problem and sketch a sparse LDL factorization based implementation. However, we prefer the Ricatti recursion implementation since it also yields feedback gain matrices, using which DDP-style trajectory updates may also be performed and the final solution can be associated with feedback control policies for closed loop execution.

Preprocessing Phase: Prior to starting ADMM updates, we execute the following backward pass to preprocess and cache the following quantities. These updates follow from standard dynamic programming arguments, based on recursively computing expressions for quadratic value functions.

- Set $\mathbf{P}_T = \mathbf{Q}_T + \rho\mathbf{I}_n$
- for $t = (T-1)\dots 0$ (Backward pass)

$$\mathbf{G}_t = [\mathbf{R}_t + \rho\mathbf{I}_m + \mathbf{B}_t^T\mathbf{P}_{t+1}\mathbf{B}_t]^{-1} \quad (26)$$

$$\mathbf{K}_t = -\mathbf{G}_t(\mathbf{B}_t^T\mathbf{P}_{t+1}\mathbf{A}_t) \quad (27)$$

$$\mathbf{U}_t = \mathbf{A}_t + \mathbf{B}_t\mathbf{K}_t \quad (28)$$

$$\mathbf{P}_t = \mathbf{Q_t} + \rho\mathbf{I}_n + \mathbf{A}_t^T\mathbf{P}_{t+1}\mathbf{U}_t + \mathbf{M}_t^T\mathbf{K}_t \quad (29)$$

$$\mathbf{f}_t = \mathbf{B}_t^T\mathbf{P}_{t+1}\mathbf{c}_t + \mathbf{r}_t \quad (30)$$

$$\mathbf{g}_t = \mathbf{q}_t + \mathbf{A}_t^T\mathbf{P}_{t+1}\mathbf{c}_t + \mathbf{K}_t^T\mathbf{f}_t \quad (31)$$

$$(32)$$

Efficient Proximal Updates: For a given input to the proximal operator $\hat{\tau}$ with control and state components $\hat{\mathbf{u}}_t, \hat{\mathbf{x}}_t$, we compute:

- Set $\mathbf{p} = \mathbf{q}_T - \rho\hat{\mathbf{x}}_T$
- for $t = (T-1)\dots 0$ (Backward pass)

$$\mathbf{k}_t = -\mathbf{G}_t(\mathbf{f}_t + \mathbf{B}_t^T\mathbf{p} - \rho\bar{\mathbf{u}}_t)$$

$$\mathbf{p} = \mathbf{g}_t - \rho\hat{\mathbf{x}}_t - \rho\mathbf{K}_t^T\hat{\mathbf{u}}_t + \mathbf{U}_t^T\mathbf{p} \quad (33)$$

The $\mathbf{k}_t$'s are stored, and then the following forward pass yields the optimal controls,

$$\mathbf{u}_t^* = \mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t \quad (34)$$

$$\mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{c}_t \quad (35)$$

The complexity of this proximal operator is $T(3mn + m^2)$ flops. Note that the above computations may be performed after a change of variables $\mathbf{x}_t \coloneqq \mathbf{x}_t - \mathbf{x}_t^k$ with respect to the current trajectory maintained in the outer loop of TROSS.

DDP-style updates: Upon convergence of the inner ADMM solve, the primal trajectory associated with the time-varying LQR is accompanied by the gain matrices $\mathbf{K}_t, \mathbf{k}_t$. Instead of propagating states and controls through linearized dynamics in Eqn. 35, we can propagate them through the original nonlinear dynamics as done in DDP, to update the trajectory in the outer loop of TROSS. This can yield faster overall convergence.

Initial State Optimization: If the initial state $\mathbf{x}_0$ is part of the optimization, then after computing the last $\mathbf{p}$ corresponding to $t = 0$ in the backward pass, we optimize the final quadratic value function $V_0(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{p}^T\mathbf{x}$. Hence,

$$\mathbf{x}_0^\star = \underset{\mathbf{x}}{\operatorname{argmin}}\, V_0(\mathbf{x}) = -\mathbf{P}^{-1}\mathbf{p} \tag{36}$$

where $\mathbf{P}, \mathbf{p}$ are the final values of these variables during caching and inference phases respectively.

B. Proximal Operators for Non-Quadratic and Non-separable Costs

For cost functions that are weighted sum of stagewise non-quadratic costs of the form, i.e.,

$$c(\tau) = \sum_{t=0}^{T-1} \alpha_t c_t(\mathbf{u}_t) + \sum_{t=0}^{T} \beta_t c_t'(\mathbf{x}_t),$$

where α_t, β_t are weights, the proximal (abbreviated prox) operator can be assembled by computing the proximal operators for each cost term independently and in parallel,

$$\texttt{prox}_{\rho,c}[\hat{\tau}] = [\texttt{prox}_{\frac{\rho}{\alpha_0},c_0}[\hat{\mathbf{u}}_0]; \ldots \texttt{prox}_{\frac{\rho}{\beta_T},c_T'}[\hat{\mathbf{x}}_T]]$$

Certain non-time-separable costs may also admit an efficient proximal operator. We discuss some cases of interest for trajectory optimization below, and point the reader to [19] for broader overview of proximal operator computations.

- *Exploiting Passive Dynamics*: Passive dynamics of the system can be exploited by encouraging the optimizer to use sparse controls [17]. Control sparsity can be enforced via l_1 norm, $c_t(\mathbf{u}_t) = \|\mathbf{u}_t\|_1$ whose prox operator is soft-thresholding, $\texttt{prox}_{\rho,\|\cdot\|_1}[\hat{\mathbf{u}}] = \max\left(\hat{\mathbf{u}} - \rho^{-1}, 0\right) - \max\left(-\mathbf{u} - \rho^{-1}, 0\right)$.
- *Path Length and Curvature*: Let $pl(\mathbf{x}_{1:T}) = \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2^2$ denote the path length cost over the state sequence. Its proximal operator is given by, $\texttt{prox}_{\rho,pl}[\hat{\mathbf{x}}_{1:T}] = \mathbf{T}^{-1}[\mathbf{x}_0;\ \hat{\mathbf{x}}_{1:T}^T]$ where $\mathbf{T}$ is a $(T+1)\times(T+1)$ banded matrix with diagonal blocks $\mathbf{T}_{ii} = (2+\frac{\rho}{2})\mathbf{I},\ \ i = 0 \ldots (T-1)$ and $\mathbf{T}_{T,T} = (1+\frac{\rho}{2})\mathbf{I}$; and the blocks above and below the main diagonal, $\mathbf{T}_{i,i+1}$ and $\mathbf{T}_{i+1,i}$, equal to $-\mathbf{I}$. In addition to path length, the planning of smooth paths with bounded curvature arises in many settings [5]. The second order difference cost associated with the Hodrick-Prescott (HP) trend filter, $hp(\mathbf{x}_{1:T}) = \sum_{t=1}^{T-1} \|\mathbf{x}_{t-1} - 2\mathbf{x}_t + \mathbf{x}_{t+1}\|_2^2$, is a natural measure of curvature of a discrete state sequence. This cost is zero if and only if the states evolve along a linear path in the state space. The proximal operator is exactly the HP filter computation for each state dimension independently, which can be compactly expressed as,

$$\texttt{prox}_{\rho,hp}[\hat{\mathbf{x}}_{1:T}] = \text{vec}\left([\hat{\mathbf{x}}_1 \ldots \hat{\mathbf{x}}_T](\mathbf{I} + \frac{2}{\rho}\mathbf{D}\mathbf{D}^T)^{-1}\right)$$

  where $\mathbf{D}$ is the Toeplitz second-order difference matrix, defined by its first row $[1\ \ -2\ 1\ 0 \ldots 0]$.
- *Constrained Robust State Estimation*: TROSS is applicable also to state estimation settings where the dynamics is perturbed by a disturbance sequence $\mathbf{w}_t$, i.e., $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t$. Likewise, sensor measurements, assumed to be nonlinear functions of state, are perturbed with a noise sequence $\mathbf{v}_t$, i.e., $\mathbf{y}_t = h(\mathbf{x}_t) + \mathbf{v}_t$. Both perturbation sequences may be subject to physical constraints. For a given fixed control sequence, $\mathbf{u}_t$, we get a constrained maximum likelihood problem that can be interpreted as an optimal control problem over $\mathbf{x}_t$ and $\mathbf{w}_t$. Robustness to measurement errors may be modeled via heavy-tailed distributions, or non-convex loss functions with bounded influence such as capped-l_1 loss which also admits an exact proximal operator despite non-convexity [11].

C. State/Control limits, and Trust Region Constraints

The projection of $\hat{\tau}$ onto the bound constraints $\mathcal{B}^k$ is simply given entrywise by $max(\tau_i^k - \mu_i^k, \tau_{min,i})$ if $\hat{\tau}_i < max(\tau_i^k - \mu^k, \tau_{min,i})$, and by $min(\tau_i^k + \mu_k, \tau_{max,i})$ if $\hat{\tau}_i > min(\tau_i^k + \mu_k, \tau_{max,i})$, and by $\hat{\tau}_i$ otherwise.

D. Safety Constraints

Suppose the constraint set $\mathcal{C}$ over trajectories is described by a set of K inequalities $g(\mathbf{x}_t) \leq 0, t = 1 \ldots T$ certifying safety of the state $\mathbf{x}_t$ at time t, where $g : \mathbb{R}^n \mapsto \mathbb{R}^K$. For example, g might measure a notion of safety margin with respect to K obstacles. The projection problem, $\text{argmin}_{\mathbf{u}_t,\mathbf{x}_t} \sum_{t=0}^{T-1} \|\mathbf{u}_t - \hat{\mathbf{u}}_t\|_2^2 + \sum_{t=1}^{T} \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2^2$ subject to $g(\mathbf{x}_t) \leq 0$ reduces to the solution $\hat{\mathbf{u}}_t$ for the controls, and separate projection problems, $\text{argmin}_{\mathbf{x}_t : g(\mathbf{x}_t) \leq 0} \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2^2$ for each time-step independently. Upon linearization at $\mathbf{x}_t^k$ (the state at time t for the trajectory τ^k), we obtain the polyhedral constraint set of the form in Eqn. 15. Note that this linearization is done in the outer loop. The inner ADMM solver simply projects onto this polyhedral set to locally bring states within safe margin from the obstacles. This projection problem is a QP which can be solved efficiently in the dual by caching the $K \times K$ gram matrix $\frac{\partial g}{\partial \mathbf{x}}(\mathbf{x}_t^k)\frac{\partial g}{\partial \mathbf{x}}(\mathbf{x}_t^k)^T$ (see [19], section 6.2). Such safety constraints can be setup by bounding the system and its environment in tight fitting volumes and computing geometric measures of separation and penetration between them. We omit details for lack of space, but demonstrate an example with ellipsoidal bounding volumes in Section IV. Our solver can also be interfaced with collision checkers in physics engines like Bullet [1].

Another approach to obstacle avoidance is to decompose the environment into a union of convex free-space regions via techniques like IRIS [3]. A finite union of convex regions is in general nonconvex, but admits exact projection

by projecting onto each region separately and taking the minimum.

E. Trust-region Adaptation via Filters

Constrained optimizers like TROSS attempt to make progress along two goals: cost reduction and constraint satisfaction. Filter methods [9], initially developed in the context of SQP techniques as a practical alternative to penalty based merit functions, track these two objectives separately and use the concept of domination from multi-objective optimization to maintain a Pareto frontier. If the inner ADMM yields a trajectory update whose cost and constraint satisfaction is acceptable to the filter (i.e., it is better than past iterates in either respect), the update step is accepted, the filter is updated, and the trust region is expanded for the next subproblem. Otherwise, the step is rejected, the trust region is contracted, and the current problem is resolved. Note that in this case, the preprocessing for TV-LQR proximal operator (subsection A) can be reused.

IV. NUMERICAL RESULTS

We now study the numerical behaviour of TROSS on three constrained nonlinear optimal control problems.

A. Car Parking with Control Limits

In this task, studied in [22], a car is described by a 4-dimensional state vector: $\mathbf{x} = (p_x, p_y, \theta, v)$ where (p_x, p_y) is the position of a point midway between the back wheels, θ is the angle of the car relative to the x-axis, and v is the velocity of the front wheels. The front wheel angle ω and the front wheel acceleration a are the control inputs, with limits ± 0.5 radians, and ± 2.0 meters-per-second-square respectively. The car needs to be maneuvered from a starting state $x = 1, y = 1, \theta = \frac{3\pi}{2}, v = 0$ to a parking goal state, $x = 0, y = 0, \theta = 0, v = 0$. The discrete-time dynamics of this system is given by,

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{pmatrix} b(v,\omega)cos(\theta) \\ b(v,\omega)sin(\theta) \\ sin^{-1}\left(sin(\omega)f(v)d^{-1}\right) \\ ha \end{pmatrix}$$

where $d = 2.0$ is the distance between the front and back axles; $f(v) = hv$ and $b(v,\omega) = f(v)cos(\omega) + d - \sqrt{d^2 - f(v)^2 sin^2(\omega)}$ (with Euler integration step size $h = 0.03$) are the rolling distances of the front and back wheel respectively. To be able to use iLQR with a robust state cost function, Tassa et. al. [22] consider a twice differentiable approximation to the Huber loss, the so-called pseudo-Huber loss: $h(x;p) = \sqrt{x^2 + p^2} - p$, for a given parameter p. The pseudo-Huber loss resembles a quadratic in a p-neighborhood around the origin, and is nearly linear thereafter. Figure 1 shows various non-smooth loss functions typically used in the robust statistics [12]. In this problem, the non-terminal state cost is $c_t(\mathbf{x}) = h(x_1;0.1)+h(x_2;0.1), t < T$, the terminal state cost is $c_T(\mathbf{x}) = h(x_1;0.1)+h(x_2;0.1)+h(x_3;0.01)+h(x_4;1.0)$, and the control cost is $\frac{1}{2}\mathbf{u}_t^T\mathbf{R}\mathbf{u}_t$ for $\mathbf{R} = diag(0.01, 10^{-4})$. TROSS is run with $\rho = 0.01$ and

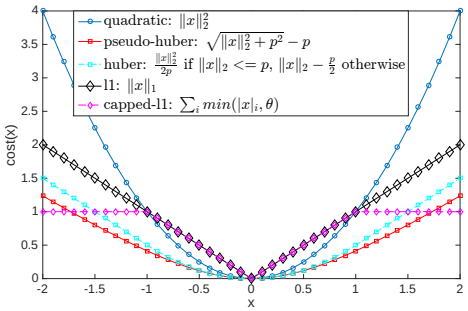

Fig. 1. Robust cost functions. Unlike iLQR, TROSS can directly optimize non-differentiable cost functions.

DDP-style updates, for 50 outer iterations each with just 5 ADMM inner iterations. The initial state and control trust region radii are set to 1.0, with maximum allowed values of 8.0 and 2.0 respectively. The time horizon is $T = 500$. The trust region expansion and shrinkage factors are 2.0 and 0.5 respectively. The controls are initialized randomly by drawing front wheel angles and accelerations independently from a Gaussian distribution with variance 0.01. The same initial controls are used to initialize iLQR, and MATLAB fmincon's SQP and active-set based nonlinear programming solvers. We used a publicly available iLQR implementation provided by the authors of [22]. Results are in Figures 2 and 3. The following observations can be made.

- As a function of both iterations and time, TROSS reduces cost faster than iLQR iterations, see Figure 3. Both iLQR and TROSS are significantly faster than general purpose nonlinear programming solvers, which took more time for a single iteration than iLQR and TROSS take to solve the entire problem.
- Both iLQR and TROSS reach a final cost of 1.905 on this problem. However, their optimal car parking maneuvers, as shown in left and middle plots in Figure 2, are very different suggesting that the optimizers follow different descent paths in the cost landscape. The TROSS optimal controls, shown in the right plot of Figure 2, respect the control limits as desired.
- Interestingly, solving the ADMM subproblem more exactly by increasing the number of inner iterations (e.g., to 10) slowed the cost reduction rate per iteration.
- The right plot of Figure 3 shows how the state and control trust region radius varies during the TROSS optimization. TROSS shrinks the trust region in the middle stages of the optimization. Also shown is how the iLQR stepsize, as found by backtracking line search, varies across its iterations: very small steps are taken in the middle stages of the optimization. Overall, TROSS makes faster progress.

Both iLQR and TROSS linearize the dynamics and build a quadratic approximation to the cost function in the neighborhood of the current trajectory in exactly the same way. However, they differ in the following respects. The Linear Quadratic Regulator (LQR) subproblem is solved exactly in iLQR yielding a search direction along which backtracking line search is performed. By contrast, TROSS solves a

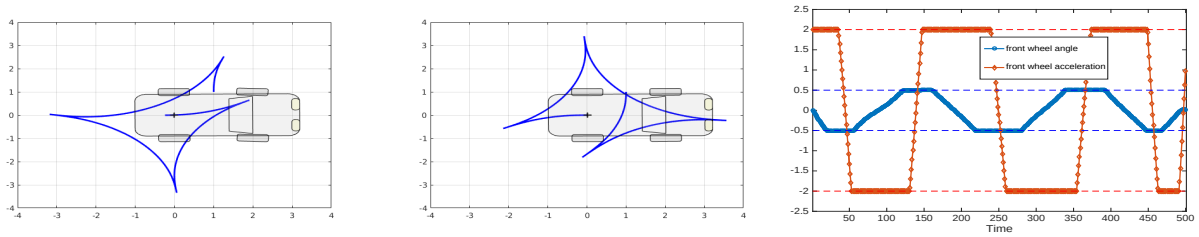

Fig. 2. iLQR (left) and TROSS (middle) Car Parking Trajectories; TROSS Optimal controls (right) respecting operational limits.

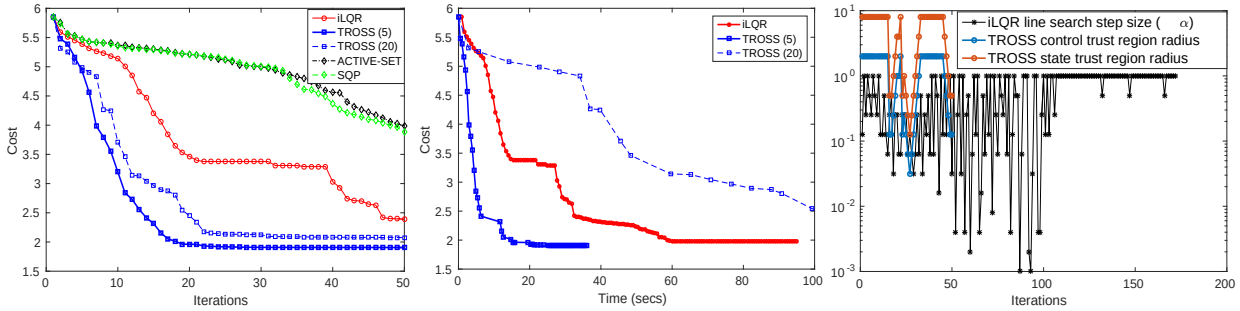

Fig. 3. Cost function improvement with respect to iterations (left) and time (middle); Trust region adaptation (right) during optimization.

constrained LQR subproblem setup within a trust region around the current iterate. This subproblem need not be solved exactly; just a few ADMM iterations suffice for yielding an improved iterate. Note that in the car-parking problem, a quadratic approximation to the pseudo-Huber loss function is accurate only in a small neighborhood of the origin which TROSS explicitly attempts to respect.

B. Dubin's Car Navigation: Obstacle Avoidance and Passive Dynamics

The Dubin's [6] car is a classic nonlinear system studied in optimal motion planning and control of mobile robots. Here, we consider a Dubin-like vehicle whose dynamics is given by,

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x},\mathbf{u}) = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -v\ \sin(\theta) \\ v\ \cos(\theta) \\ u \end{pmatrix}$$

where x, y is the two-dimensional position of the vehicle in the environment, $v = 1$ meters per second is the fixed speed of the vehicle, θ is the yaw angle (angle wrt y-axis), and u is the control input. In other words, the car moves at constant velocity and control can only influence the orientation. Thus its zero-control passive dynamics can be exploited to make progress towards the goal. We discretize the dynamics using Euler's method with a step size of 0.02 and control is applied for 0.1 seconds. The task is to navigate the car from position $x = -3, y = -3$ and pose $\theta = 0$, to the goal $x = 9, y = 9$ returning back to the pose $\theta = 0$, in a time horizon of 10 seconds corresponding to $T = 100$. The body of the car is modeled by fitting a minimum volume ellipsoid to the points $(\pm 1, 0), (0, -2), (0, 1)$. The environment comprises of 3 obstacles with ellpsoidal bounding volumes, $a(x - c_x)^2 + b(y - c_y)^2 \leq 1$, centered at locations $(c_x, c_y) = (0, 5), (5, -2.5)$ and $(0, -0.5)$, with semi-axes lengths $(a, b) = (0.03, 0.09), (0.6, 0.2), (0.4, 0.2)$ respectively. The task is to minimize path length plus l_1 norm of controls, $\sum_{t=0}^{T-1} \|x_{t+1} - x_t\|_2^2 + \beta\|\mathbf{u}\|_1$ subject to vehicle dynamics, goal constraints, and obstacle avoidance with a user-specified safety margin. The shortest path navigation requires the vehicle to curve around the obstacle until its pose changes by 90 degrees, and then reorient itself near the goal to be back at $\theta = 0$. The l_1 penalty term on control inputs encourages controls from being turned off if progress towards the goal can be made by relying on passive dynamics of the system. The trajectories and associated sparse controls estimated by TROSS on this problem are shown next. It can be seen that control inputs are only used for the initial turn and then to curve (bold shaded) just enough around the obstacle. The "cruise-control" part of the trajectory is piecewise linear (lightly shaded).

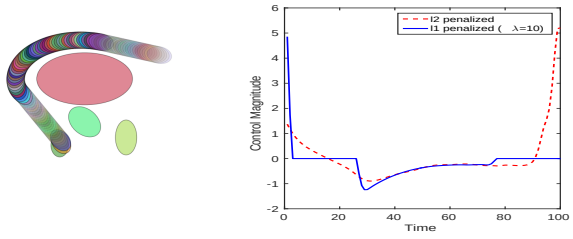

Fig. 4. Cruise Control: (left) The control is switched off for lightly shaded segments. (right) control inputs with or without sparsity constraints.

C. Robust State Estimation

Here we consider a benchmark robust filtering problem described in [14] involving tracking the trajectory of a body falling through an atmosphere with exponentially decaying density and uncertain aerodynamics, based on corrupted

radar measurements. Ignoring gravitational acceleration, the discrete-time dynamics is written as,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t + h \begin{pmatrix} -x_2 \\ -\gamma x_2^2 e^{-\eta x_1} \end{pmatrix} + \mathbf{u}_t$$

where γ, η a constants; the state vector comprises of the altitude x_1, and downward velocity x_2, and variables $\mathbf{u} \in \mathbb{R}^2$ encode additive dynamics noise. The task is to estimate the trajectory of the body from noisy radar distance measurements $y_t = g(\mathbf{x}_t) = \sqrt{b^2 + (x_{1t} - a)^2}$ where (b, a) are coordinates of the radar (see [14] for details). We consider a joint state evolution and measurement model whose discrete-time dynamics evolves as,

$$\tilde{\mathbf{x}}_{t+1} = \begin{pmatrix} \mathbf{x}_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} f(\mathbf{x}_t, \mathbf{u}_t) \\ g(f(\mathbf{x}_t, \mathbf{u}_t)) \end{pmatrix}.$$

For a sequence of measurements $\hat{y}_t$, we consider the tracking cost $c_t(\tilde{\mathbf{x}}_t) = 0.01\hat{l}_1(\tilde{x}_3 - \hat{y}_t)$ where $\hat{l}_1(x, \theta) = max(|x|, \theta)$ is chosen to be the capped-l_1 loss which is non-smooth and non-convex (see Figure 1) and caps the influence of a measurement residual at 0.25. Despite non-convexity, the capped-l_1 loss admits an exact proximal operator [11]. We impose l_1 penalty on dynamics noise. A trajectory of length $T = 600$ ($h = 0.1$) is generated from the system by perturbing states and measurements with random non-Gaussian disturbances drawn from a half normal distribution $|\mathcal{N}(0, 0.01)|$ and truncated normal distribution $\max(\mathcal{N}(0, 1.0), 0)$ respectively. Around 40% of the trajectory is subject to such noise. We run `TROSS` with $\rho = 1$, 40 outer and 25 inner iterations, with initial state and control trust region radius set to 1.0. Non-negativity constraints are also imposed on the state variables. State estimation comparisons against a tuned Extended Kalman Filter (EKF), which makes strong Gaussianity assumptions on dynamics and measurement noise, are shown in Figure 5. An EKF-based forward-backward smoother also gives similar results. `TROSS` with capped-l1 loss effectively ignores outlier measurements and provides better state estimates.

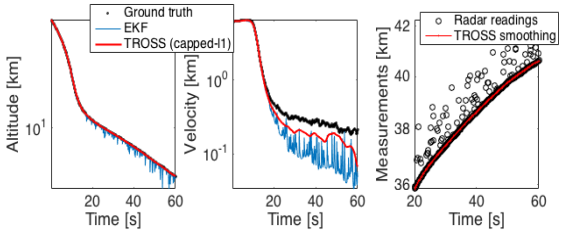

Fig. 5. Robust State Estimation

V. Conclusions and Future Work

Sequential splitting is an effective approach for highly constrained nonlinear control problems. Its application to MPC, moving horizon estimation and guided policy search [15], together with benchmarks against SQP methods and existing trajectory optimizers are avenues for future developments.

Acknowledgements

We thank Alexander Herzog, Sergey Levine, Benjamin Recht, Katya Scheinberg, Yuval Tassa and Vincent Vanhoucke for helpful technical discussions.

References

[1] Bullet Physics Library. `http://bulletphysics.org`.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3, 2011.

[3] R. Deits and R. Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. *Workshop on the Algorithmic Fundamentals of Robotics,*, 2014.

[4] S. Diamond, R. Takapoui, and S. Boyd. A general system for heuristic solution of convex problems over nonconvex sets. *arXiv preprint arXiv:1601.07277*, 2016.

[5] Y. Duan, S. Patil, J. Schulman, K. Goldberg, and P. Abbeel. Planning locally optimal, curvature-constrained trajectories in 3d using sequential convex optimization. *ICRA*, 2014.

[6] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79, 1957.

[7] J. C. Dunn and D. P. Bertsekas. Efficient dynamic programming implementations of newton's method for unconstrained optimal control problems. *JOURNAL OF OPTIMIZATION THEORY AND APPLICATION*, 63(1), 2013.

[8] M. M. F. Borrelli, A. Bemporad. *Predictive Control for linear and hybrid systems*. http://www.mpc.berkeley.edu/mpc-course-material, 2015.

[9] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 2002.

[10] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Rev*, 47(1), 2006.

[11] P. Gong, C. Zhang, Z. Lu, J. Z. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. *International Conference on Machine Learning (ICML)*, 2013.

[12] P. Huber and E. M. Ronchetti. *Robust Statistics*. Wiley (2nd Edition), 2009.

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

[14] C. Karlgaard and H. Schaub. Comparison of several nonlinear filters for a benchmark tracking problem. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.

[15] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17(39), 2016.

[16] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. *International Conference on Informatics in Control, Automation and Robotics*, 2004.

[17] M. Nagahara, D. E. Quevedo, and D. Nesic. Maximum hands-off control: A paradigm for control effort minimization. *IEEE Transactions on Automatic Control*, 16(4), 2016.

[18] B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Trans. on Control Systems Tech.*, Nov 2013.

[19] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Machine Learning*, 1, 2014.

[20] A. U. Raghunathan and S. D. Cairano. Infeasibility detection in alternating direction method of multipliers for convex quadratic programs. *IEEE 53rd Annual Conference on Decision and Control (CDC)*, 2014.

[21] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, S. P. Henry Bradlow, Jia Pa and, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research*, June 2014.

[22] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. *International Conference on Robotics and Automation (ICRA)*, 2014.

[23] M. J. TENNY, S. J. WRIGHT, and J. B. RAWLINGS. Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming. *Computational Optimization and Applications*, 28(1), 2004.

[24] Y. Wang, W. Yin, and J. Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *UCLA CAM Report 15-62 (http://arxiv.org/abs/1511.06324)*, 2016.