Sparsity-Inducing Optimal Control via Differential Dynamic Programming

Traiko Dinev*, Wolfgang Merkt*, Vladimir Ivan, Ioannis Havoutis, Sethu Vijayakumar

***Abstract*— Optimal control is a popular approach to synthesize highly dynamic motion. Commonly, L_2 regularization is used on the control inputs in order to minimize energy used and to ensure smoothness of the control inputs. However, for some systems, such as satellites, the control needs to be applied in sparse bursts due to how the propulsion system operates. In this paper, we study approaches to induce sparsity in optimal control solutions—namely via smooth L_1 and Huber regularization penalties. We apply these loss terms to state-of-the-art Differential Dynamic Programming (DDP)-based solvers to create a family of sparsity-inducing optimal control methods. We analyze and compare the effect of the different losses on inducing sparsity, their numerical conditioning, their impact on convergence, and discuss hyperparameter settings. We demonstrate our method in simulation and hardware experiments on canonical dynamics systems, control of satellites, and the NASA Valkyrie humanoid robot. We provide an implementation of our method and all examples for reproducibility on GitHub.**

I. Introduction

The propulsion systems of orbital satellites have a unique control limitation. In many cases, they use impulsive cold gas or bi-propellant thrusters incapable of or ineffective at low rates of firing. The resulting control then relies on fewer longer bursts of thrust to generate a constant amount of force over time while keeping the thrusters off between the bursts.

The required control inputs are then a sequence of binary on/off commands that are activated sparsely throughout the motion [1]. We call this type of control *sparsity-inducing*, referring to the sparse use of control inputs throughout the trajectory. Such control will prefer zero control (off) followed by a high control action (on) to continuous corrective commands applied throughout the whole trajectory.

Selecting the required degrees of freedoms (DoFs) of a redundant system such as a humanoid robot is another application of *sparsity-inducing* control. Instead of deactivating the control inputs, the planner deactivates unnecessary joints. Consider a reaching task for the 38-DoF Valkyrie humanoid robot, where the goal is to extend the hand (end-effector) forward to point at a target. Solving this via motion planning involves finding suitable control inputs for the entire humanoid such that it balances itself and extends the hand.

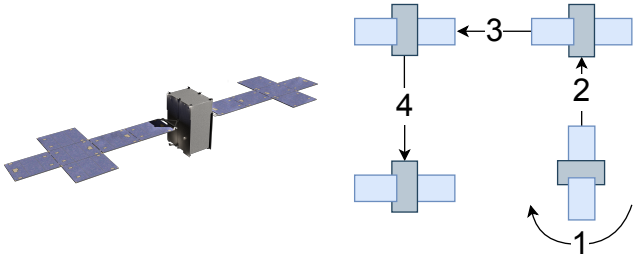


Fig. 1: SSL-1300 satellite model and satellite maneuver.

With traditional control-penalty methods, the planner readily discovers a motion where all the joints are simultaneously moving (cf. Fig. 10). Such a motion is arguably unnecessary and in fact undesirable as it requires more complex control to coordinate the motion. This can result in trajectories that are more difficult to execute and track by the low-level controller and may require more energy.

To achieve sparsity in the controls as well as automatic joint selection, we propose to use a sparsity-inducing penalty term in the control cost. This serves to both switch off unnecessarily small control inputs in the case of high-DoF robots and to discover thruster-like behavior for satellites.

A. Related Work

Optimal control methods synthesize dynamically consistent motion satisfying a set of task and dynamics constraints while minimizing an optimality criterion. Shooting methods, which optimize over the control inputs, and in particular algorithms derived from Differential Dynamic Programming (DDP) [2], have recently received renewed interest [3]–[5]. In contrast to simultaneous transcription methods [6], [7], shooting methods have faster computation times by explicitly exploiting the temporal structure and implicitly enforcing the dynamic feasibility of the solution.

A common optimality criterion is energy optimization, which is traditionally a squared cost on the control inputs (L_2 norm regularization). In practice, this leads to smooth control profiles and has been widely applied to canonical dynamic systems, computation of flight trajectories, as well as to synthesize highly dynamic maneuvers for legged robots [4], [5]. However, as no sparsity is introduced, on redundant systems this frequently leads to moving many joints even if not all joints are required to complete the task (cf. Fig. 10).

Sparsity in control inputs for planning was studied in the context of satellite motion planning [8]. There the authors used the L_1 norm, also known as *Lasso* model. Similarly, the authors in [9] applied an L_1 penalty using

* Equal contribution, alphabetical order.

Wolfgang Merkt and Ioannis Havoutis are with the Oxford Robotics Institute, University of Oxford, UK.

Traiko Dinev, Vladimir Ivan, and Sethu Vijayakumar are with the Edinburgh Centre for Robotics, The University of Edinburgh, UK.

This research was supported by (1) the European Commission under the Horizon 2020 project Memory of Motion (MEMMO, ID: 780684) and (2) the Engineering and Physical Sciences Research Council (EPSRC) UK RAI Hubs for Offshore Robotics for Certification of Assets (ORCA, EP/R026173/1) and Future AI and Robotics for Space (FAIR-SPACE, EP/R026092/1).

an Alternating Direction Method of Multipliers (ADMM) approach separating the problem into an optimal control update and a soft thresholding update. Whereas previous work considered an L_1 penalty applied to the force at the center of mass of the satellite, here we model the thruster behavior directly. Finally, the authors in [1] obtained thruster controls by optimizing the timing of thruster pulses, which are modeled as on-off controls. Here we use smooth L_1 costs to penalize the otherwise continuous thruster forces. As a result, our approach is directly applicable in standard optimal control frameworks.

Recently, the concept of sparsity has gained additional attention in the trajectory optimization and control community for terrestrial/traditional robotics applications such as manipulation. [10], for instance, investigated the use of Mixed-Integer and Lasso regression to reduce joint motion on humanoid robots in a hierarchical inverse dynamics control scheme. Nonetheless, enforcing sparsity for planning over longer horizons continues to be a challenge.

It is well known in the machine learning community that L_1 introduces a discontinuity at 0 that makes it non-differentiable [11]. Several differentiable metrics have been proposed to deal with this issue [9], [12]. In this paper, we study two such costs when applied to DDP: the SmoothL1 [13] and Huber differentiable approximations [12] to L_1. Using such sparsity-inducing cost terms with DDP raises two challenges requiring careful consideration: i) ensuring numerical stability/conditioning as efficient implementations assume positive-definiteness of the control Hessian, and ii) trading off achievements of desired tasks with sparsity of solutions through control regularization. Note that these challenges do not arise when using direct transcription/collocation where tasks are enforced with hard constraints and solved using off-the-shelf Non-linear Programming (NLP) solvers.

B. Contributions

We study approaches to induce sparsity in optimal control solutions and make the following contributions:

1) Introduce sparsity-inducing regularization terms in DDP-type solvers.
2) Compare different strategies for sparsity-inducing regularization, namely SmoothL1, Huber, and Pseudo-Huber loss, in terms of their convergence and numerical stability.
3) Demonstrate our approach for the swing-up of a cart-pole and for satellite thruster control. Additionally, we demonstrate hardware manipulation experiments using the Valkyrie humanoid robot.

We provide our implementation and evaluations as open source software for reproduction.[1] A supplementary video is available at `https://youtu.be/YMXRZjFsqhc`.

II. Control Regularization

We begin by reviewing the use of L_2 penalties in the optimal control literature. The L_2 loss is defined as:

$$L_2(x) \triangleq x^2.$$

[1] `https://github.com/ipab-slmc/sparse_ddp`

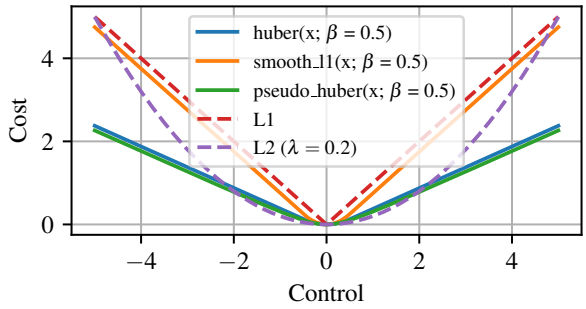


Fig. 2: Comparison of the different regularization schemes for the control cost using a range of hyper-parameters: L_2, L_1, $L_{\text{1-smooth}}$, L_{Huber}, and $L_{\text{pseudo-Huber}}$.

The L_2 loss is used to regularize solutions by penalizing large positive or negative control inputs in the optimal control setting or features in machine learning.

In contrast, the L_1 loss is used to penalize solutions for sparsity, and as such, it is commonly used for feature selection in the machine learning community [14]. The L_1 loss is defined as the absolute value of its argument:

$$L_1(x) \triangleq |x|$$

When L_1 is used with gradient-based optimizers as a regularization, it drives its argument to exactly 0 as opposed to small values. This is explained by the condition that the gradients of the regularization parameter and the task cost must be parallel.

Using the L_1 loss directly in gradient-based optimization is difficult due to the discontinuity at $x = 0$ where the gradient is undefined. Smooth approximations to the L_1 function can be used in place of the true L_1 penalty. In this paper, we consider a smooth L_1 function from [13], which combines L_1 and L_2 losses, defined as:

$$L_{\text{smooth-l1}}(x) = \begin{cases} 0.5x^2/\beta & \text{if } |x| \leq \beta \\ |x| - 0.5\beta & \text{otherwise} \end{cases} \tag{1}$$

where β defines where the function switches from an L_1 to an L_2 cost. For small $x \leq \beta$, *SmoothL1* switches to L_2, since L_2 has a gradient at 0.

We study another variant of combining L_1 and L_2 regularization, namely the Huber loss [12]:

$$L_{\text{huber}}(x) = \begin{cases} 0.5x^2 & \text{if } |x| \leq \beta \\ \beta(|x| - 0.5\beta) & \text{otherwise} \end{cases} \tag{2}$$

where β is again a shape parameter. The Huber loss has a variable slope, controlled by β in addition to mixing L_1 and L_2. This can be seen in Fig. 2, where for $\beta = 0.5$ the Huber cost has a lower slope than SmoothL1.

Finally, we consider a smooth approximation to the Huber loss, the Pseudo-Huber loss, as defined in [15, Appendix 6]:

$$L_{\text{pseudo-huber}}(x) = \beta^2 \left(\sqrt{(1 + \frac{x}{\beta}^2)} - 1 \right). \tag{3}$$

We illustrate the considered losses for different settings of their hyper-parameters in Fig. 2. The shape parameters of

the smooth variants control how closely they approximate the true L_1 and Huber losses, respectively. The choice of the control regularization and its parametrization has an impact on convergence and sparsity of the output of the optimal control formulation.

III. Optimal Control

We consider the robot as a dynamic system described by state $\boldsymbol{x}$ composed of generalized coordinates $\boldsymbol{q}$ and generalized velocities $\boldsymbol{v}$. The system evolves under applied control inputs $\boldsymbol{u}$ according to the state transition function $\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t)$ which incorporates the differential dynamics as well as an integration scheme. Here, we use a geometric representation of the configuration manifold of floating-base systems ($\mathbb{SE}(3)$) with its geometric integrators along with an energy-conserving symplectic integration scheme of the differential dynamics.

To describe a discrete optimal control problem with a fixed horizon, we additionally specify the integration time step Δt and time horizon T and the number of discretization knots N. This yields a state trajectory $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ and control trajectory $U = \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_{N-1}\}$. Tasks and constraints are enforced by minimizing a cost function:

$$J(X, U) = h(\boldsymbol{x}_N) + \sum_{t=1}^{N-1} l(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{4}$$

Shooting methods in particular minimize $J(\cdot)$ with respect to control inputs only:

$$U^* = \arg\min_U \; J(X, U)$$

where U^* is the optimal open-loop control trajectory. The corresponding state trajectory is obtained by performing a forward roll-out using the state transition function.

A. Differential Dynamic Programming

Differential Dynamic Programming (DDP) [2], [16] is a classical method to solve the above unconstrained optimal control problem using Bellman's principle of optimality. DDP begins by making a quadratic approximation of the action-value function Q around a reference trajectory $X^{(0)} = \{\boldsymbol{x}_1^{(0)}, \dots, \boldsymbol{x}_N^{(0)}\}$ and $U^{(0)} = \{\boldsymbol{u}_1^{(0)}, \dots, \boldsymbol{u}_{N-1}^{(0)}\}$. The Q-function is defined as:

$$Q(\boldsymbol{x}, \boldsymbol{u}, t) = l(\boldsymbol{x}, \boldsymbol{u}) + V(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), t+1) \tag{5}$$

where the value function computes the "goodness" of state $\boldsymbol{x}$, the Q-function gives the same quantity for a state and an action. DDP minimizes the second-order Taylor expansion of the Q-function:

$$\begin{aligned} Q(\boldsymbol{x}_t, \boldsymbol{u}_t) = Q(\boldsymbol{x}_t^{(0)}, \boldsymbol{u}_t^{(0)}) + Q_x \delta\boldsymbol{x}_t + Q_u \delta\boldsymbol{u}_t + \\ \frac{1}{2}\delta\boldsymbol{x}_t^T Q_{xx} \delta\boldsymbol{x}_t + \frac{1}{2}\delta\boldsymbol{u}_t^T Q_{uu} \delta\boldsymbol{u}_t + \delta\boldsymbol{u}_t^T Q_{ux} \delta\boldsymbol{x}_t \end{aligned} \tag{6}$$

where $\delta\boldsymbol{x}_t = \boldsymbol{x}_t - \boldsymbol{x}_t^{(0)}$ and $\delta\boldsymbol{u}_t = \boldsymbol{u}_t - \boldsymbol{u}_t^{(0)}$ and the subscript notation is shorthand for the partial derivative of Q evaluated at the reference trajectory point for $t \in [1, N]$. In the following, we drop the subscripts to denote way points for readability. We then give the derivatives:

$$\begin{aligned} Q_x &= l_x + \boldsymbol{f}_x^T V_x' \\ Q_u &= l_u + \boldsymbol{f}_u^T V_x' \\ Q_{xx} &= l_{xx} + \boldsymbol{f}_x^T V_{xx}' \boldsymbol{f}_x^T + V_x' \cdot \boldsymbol{f}_{xx} \\ Q_{uu} &= l_{uu} + \boldsymbol{f}_u^T V_{xx}' \boldsymbol{f}_u^T + V_x' \cdot \boldsymbol{f}_{uu} \\ Q_{ux} &= l_{ux} + \boldsymbol{f}_u^T V_{xx}' \boldsymbol{f}_x^T + V_x' \cdot \boldsymbol{f}_{ux} \end{aligned}$$

where V' is shorthand for $V(\cdot, t+1)$. The last terms are shorthand for tensor products.

DDP minimizes the quadratic approximation with respect to the new coordinate system. Hence we obtain the local feedback control law:

$$\delta\boldsymbol{u}_t^* = \arg\min_{\delta\boldsymbol{u}_t} Q(\cdot) = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x}_t)$$

with the feed-forward modification $\boldsymbol{k} = Q_{uu}^{-1} Q_u$ and state feed-back term $K = Q_{uu}^{-1} Q_{ux}$. Since $\delta\boldsymbol{u}_t^*$ is the minimum of the Q-function, DDP obtains a recursive set of equations for the value function at every time-step:

$$\begin{aligned} V = Q(\boldsymbol{x}_t, \boldsymbol{u}_t^*) = Q(\boldsymbol{x}_t^{(0)}, \boldsymbol{u}_t^{(0)}) - Q_u Q_{uu}^{-1} Q_u \\ V_x = Q_x - Q_{xu} Q_{uu}^{-1} Q_u \\ V_{xx} = Q_{xx} - Q_{xu} Q_{uu}^{-1} Q_{ux} \end{aligned}$$

In order to evaluate these equations, Q and its derivatives are evaluated at the reference trajectory state $\boldsymbol{x}_t^{(0)}$ and the optimal control $\delta\boldsymbol{u}_t^*$ as calculated above. This is performed in a backward pass from knot $t = N$ to $t = 1$. The backward pass is followed by a forward pass in order to obtain the new state sequence $\hat{X} = \{\hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_N\}$ and controls $\hat{U} = \{\hat{\boldsymbol{u}}_1, \dots, \hat{\boldsymbol{u}}_{N-1}\}$:

$$\begin{aligned} \hat{\boldsymbol{x}}_1 &= \boldsymbol{x}_1^{(0)} \\ \hat{\boldsymbol{u}}_t &= \boldsymbol{u}_t^{(0)} - \boldsymbol{k}_t - K_t(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t^{(0)}) \\ \hat{\boldsymbol{x}}_{t+1} &= \boldsymbol{f}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \end{aligned}$$

IV. Enforcing Sparsity with L_1 and Huber Costs

DDP minimizes a general cost function $J(X, U)$ of the form in (4). In this work, we propose adding an additional cost term for each control input $\boldsymbol{u}_t$ that induces sparsity. The new cost function thus becomes:

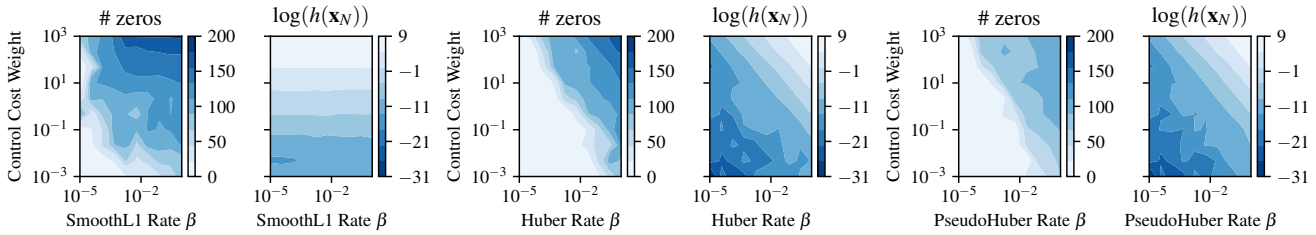
$$J(X, U) = h(\boldsymbol{x}_N) + \sum_{t=0}^{N-1} \left[l(\boldsymbol{x}_t, \boldsymbol{u}_t) + \lambda \; l_s(\boldsymbol{u}_t) \right] \tag{7}$$

where $l_s(\boldsymbol{u}_t)$ is one of the sparsity-inducing losses described in section II and λ is a strength parameter, which controls the relative effects of the regularization loss and the objective loss.

For the cartpole and satellite examples, we use quadratic state costs of the following form:

$$\begin{aligned} h(\boldsymbol{x}_N) &= (\boldsymbol{x}_N - \boldsymbol{x}^*)^T Q_f (\boldsymbol{x}_N - \boldsymbol{x}^*) \\ l(\boldsymbol{x}_t, \boldsymbol{u}_t) &= (\boldsymbol{x}_t - \boldsymbol{x}_t^*)^T Q (\boldsymbol{x}_t - \boldsymbol{x}_t^*) \end{aligned}$$

where Q is a diagonal weighting matrix for each of the NDX dimensions of $\boldsymbol{x}_t$ and the matrix Q_f is a weighting term for



(a) SmoothL1 (b) Huber (c) Pseudo Huber

Fig. 3: Comparison of different sparsity-inducing control cost terms on the cartpole system: Sparsity of solutions and final costs for a grid-search over hyper-parameters β (shape) and λ (strength). Darker color indicates more sparsity and lower final cost respectively.

$\boldsymbol{x}_N$, respectively. For the Valkyrie example, we demonstrate the use of nonlinear task cost functions such as end-effector position, stability cost, and joint limits from [17].

The parameters λ together with β are hyper-parameters and their values define the interaction between the sparsity loss and the optimization criterion (task). We study their effect on the convergence of the solution in detail using the canonical cartpole in the following section.

V. Effects of Sparsity Loss on Toy Problems

We firstly study the effects of sparsity-inducing costs on a one-dimensional problem—the swing-up of a cartpole, which is a canonical optimal control problem where a pendulum is attached to a cart moving on an infinite friction-less track. The goal is to swing the pendulum upright and move the cart to the origin. The problem is underactuated—the control inputs are linear forces on the cart, whereas the pendulum joint is not controlled. The time horizon is $T = 200$ with $\Delta t = 0.01$s, resulting in a 2 s trajectory. The control limits are ± 30N and $Q_f = 100\ \mathbb{I}_4$, where $\mathbb{I}_4$ is the 4×4 identity matrix.

A. Effects of weight and shape parameters on sparsity

Firstly, we examined the effect of the weighting term λ and shape parameter β on sparsity. β for all functions controls where the switching between an L_2 cost (for small $x < \beta$) and an L_1 cost (for $x \geq \beta$) occurs. We consider controls to be zero when they are within $[-\beta, \beta]$.

The results of a grid search over β and λ are in Fig. 3a for SmoothL1, Fig. 3b for Huber, and Fig. 3c for PseudoHuber. We plotted both the number of zeros and the final task cost—the latter tells us how close we are to the goal state.

A desired property of sparsity costs is that sparsity should increase with the weight term λ. As expected, we see a trade-off between sparsity and task cost—the more regularization, the more sparsity, however at a higher task cost. This is in fact what the grid search shows. Another important property we observe is that for lower tolerances β, a much higher weight is required to achieve sparsity. Thus we can extract a criterion for choosing sparsity—1) pick the largest β parameter according to how much noise tolerance the system has and 2) adjust the control cost weight until the desired amount of sparsity is achieved. The noise tolerance of the system is the maximum

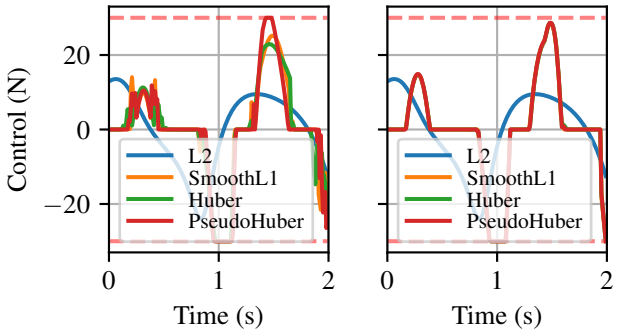


Fig. 4: Sparse control solutions for the cartpole system: Left with control artifacts and right with smooth controls.

value of controls beyond which rapid fluctuations can not be tolerated.

In the solutions for $\beta = 10^{-3}$ in Fig. 4(left), all solutions achieve a final task cost of less than 10^{-5} with 73, 58 and 86 zero controls for SmoothL1, Huber and PseudoHuber, respectively, yet we observe artifacts showing rapid control changes. Increasing the weight (from 7, 9, and 0.01 to 25, 25, and 0.023 for Huber, PseudoHuber and SmoothL1, respectively) can reduce these artifacts, while also increasing sparsity. In Fig. 4(right) the solutions have 101, 102 and 89 zero controls and produce smoother control profiles. However, this comes with an increase of final state cost from less than 10^{-5} to less than 10^{-4}.

In general, sparsity-inducing costs together with control/actuator limits produce so-called "bang-bang" control. On a real system, aggressive bang-bang control requires the actuator to change the output torque rapidly at each time-step. This is usually not possible due to actuator dynamics, for example, when using electric motors on the cart pole in our example. However, in some domains, namely satellite control, the underlying physical system and actuators are only capable of bang-bang control and this in fact is a desirable property.

VI. Thruster Control for Satellites

We now consider a satellite as a floating rigid body in $\mathbb{SE}(3)$ actuated through forces produced by two primary propulsion thrusters on the front and back of the satellite and four steering thrusters on each of its remaining four sides ($NX = 13$, $NU = 18$). We generated a center of mass

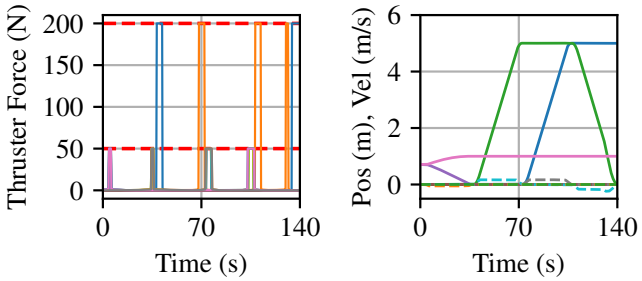


Fig. 5: Satellite thruster trajectory (SmoothL1) and corresponding state trajectory.

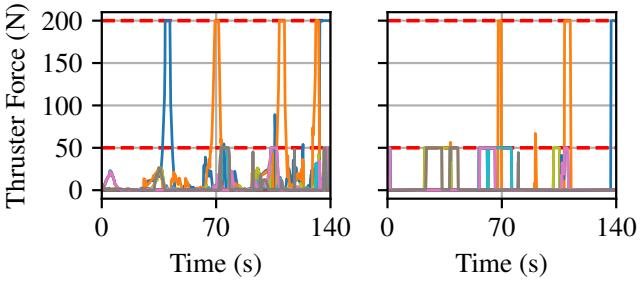


Fig. 6: Satellite control trajectories for weights 10^{-5} and 10^{-1} and a PseudoHuber loss.

tracking trajectory with 4 discrete stages solved in a single optimization problem, as illustrated in Fig. 1. The problem was discretized with $\Delta t = 0.1$ with $T = 1400$ knots for a 140-second trajectory. Thruster forces were constrained in $[0, 200]$ N for the front and back and $[0, 50]$ N for the side thrusters, since there are four for each direction.

The resulting control (thrust) and state trajectories are shown in Fig. 5. The different colors correspond to different thrusters being activated at the corresponding times. We see thruster peaks at the start and end of each of the stages and reach the corresponding set output forces. The right side of the figure shows the position and linear velocity trajectories in state space.

A. *Effects of weight parameters*

We next examine the effects of the different weights on the satellite problem. In Fig. 6 we plotted the control trajectories for different weights—$\lambda = 10^{-5}$ and $\lambda = 10^{-1}$. The plot for the correctly tuned $\lambda = 10^{-3}$ is in Fig. 5. We see a similar trend as in the cart-pole system—if the control weight is too small, we observe artifacts in solution space. Tuning the weight produces sparse solutions with no artifacts and the desired bang-bang control profile. Finally, we observe a different result when increasing the weight on the satellite example—this leads to longer thruster bursts on the thrusters with smaller limit, which in fact leads to less sparsity—23649 zero controls compared to 24509 when tuned. This can be explained by the reduction of peaks at 200 N, which are penalized more, which in turn leads to the solver compensating by switching on the 50 N thrusters.

B. *Effects on convergence*

Finally, we examine the effects on convergence for L_1 costs. A plot of the time to convergence for all considered cost terms is shown in Fig. 7. We computed this over a grid of weights $\lambda \in [10^{-5}, 10^{-1}]$ for $\beta = 1$. Generally, time to convergence is increased for all sparse costs with Huber being slowest and Pseudo-Huber fastest to converge on average. This is expected as sparsity-inducing cost terms have less steep gradients further away from zero, where the gradient for an L2 loss would be larger.

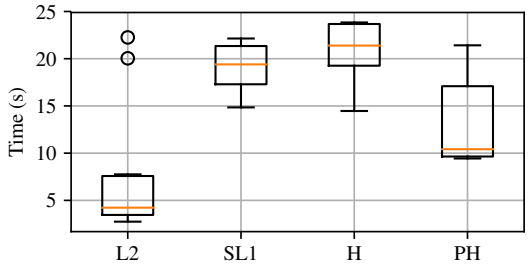


Fig. 7: Timing analysis on the satellite example: Using sparse costs leads to slower overall convergence.

VII. Active Joint Selection for Lower-Dimensional Tasks on Redundant Systems

We next applied the sparse costs to a reaching task on a 38-DoF humanoid robot. We use acceleration-based linear system dynamics and nonlinear general task costs. In this case, the three-dimensional reaching task requires only two joints to move (the shoulder and the hip). This is illustrated in Fig. 10. The goal is to reach to $\boldsymbol{x}^* = [0.5, 0.2, 0.9]$, a point directly in front of the robot at hip-height. We discretized into $T = 20$ knots with $\Delta = 0.1$ s for a 2 s trajectory.

The resulting state and control plots for L_2 are in Fig. 8. Solving the problem with L_2 control regularization produces a solution that moves more joints than necessary. This is easily seen in the corresponding state plots (Fig. 8 and 9), showing the positions and velocities of the joints that move.

Applying a sparsity cost (Pseudo-Huber) to the problem leads to the solver choosing to move only the required joints. The resulting trajectories in Fig. 9 show this clearly. However, due to the approximation of the L_1 costs, numerically the robot is not moving 3 joints, as is apparent, but rather 7 have velocities greater than 10^{-3}. Compared with L_2, which moves 26 joints, this is nonetheless a significant reduction.

Finally, we executed the motion plans on the physical robot. The trajectories are tracked using an inverse dynamics based whole-body controller. We compared a solution with an L_2 cost and a Huber cost. We plot the tracking results (distance of end-effector to target) in Fig. 11. For the Huber sparse trajectory, tracking is better both during and at the end of the trajectory.

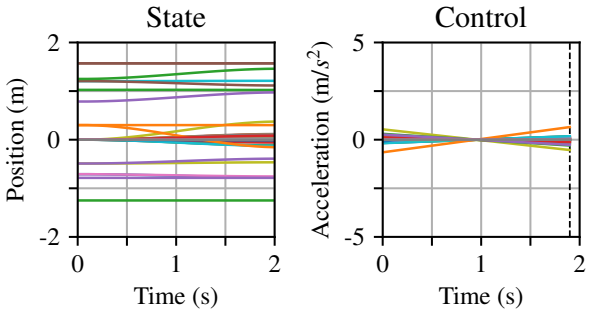


Fig. 8: Valkyrie reaching task. L_2 uses 25 joints.

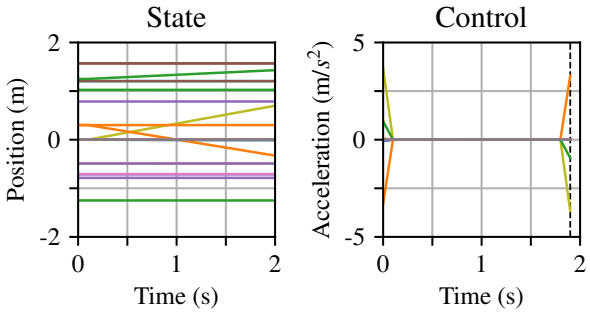


Fig. 9: Valkyrie reaching task. Pseudo Huber uses 7 joints.

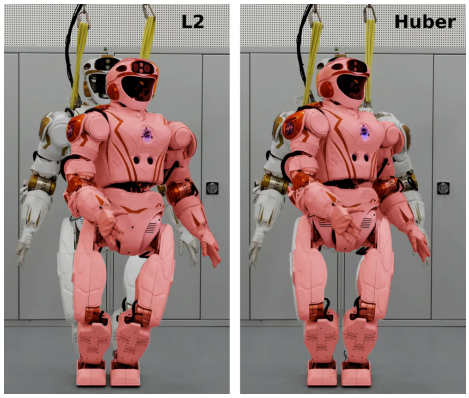


Fig. 10: Reaching to a position target: Using L_2, the robot reaches the target while moving a majority of the joints by small amounts. A Huber loss induces sparsity improving tracking and maintaining an equal contact force distribution.

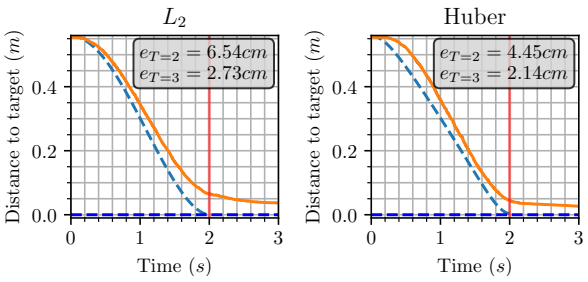


Fig. 11: Tracking results for the Valkyrie reaching task. The red line highlights the end of the commanded trajectory. We highlight the task-space error at the end of the trajectory ($T = 2$s) and the end of the experiment. This illustrates that the whole-body controller adds a delay to the motion execution, and that the tracking error is lower for the sparsity-induced trajectory.

VIII. Discussion

We studied the effects of using an L_1 cost for the control of dynamic systems using optimal control. Since L_1 is not continuously differentiable, we studied three approximations: the SmoothL1, Huber, and PseudoHuber losses.

On a simple cartpole problem, L_1 costs lead to sparsity in control space by making a subset of the controls 0 and producing peaks that resemble square waves. We analyzed the performance of L_1 costs over a grid of values for the shape parameter β, which thresholds switching between L_1 and L_2, and the control cost weight λ. For smaller values of β a much larger control weight is required to achieve sparsity. Larger control weights, however, lead to higher task costs. We thus propose picking the largest value of β according to the system's noise tolerance and then fitting the weight λ until the desired level of sparsity is achieved. We further motivate this approach by showing that relying on sparsity and final task cost alone can lead to non-smooth control trajectories with visible artifacts.

Scaling L_1 costs to real-world robots presents new challenges. We successfully applied L_1 to a kino-dynamics optimal control problem on the Valkyrie robot to select a subset of active joints for a low-dimensional reaching task. While L_2 losses use more joints than necessary, L_1 can automatically reduce the number of joints by setting the corresponding controls to 0. Sparse controls can be in practice tracked better. However, sparse controls also result in bang-bang control with higher commanded accelerations that can damage the hardware.

On the other hand, this is a desired control profile for thruster control for satellites. We were able to achieve thruster-like behavior with L_1 costs in order to track a multi-stage center of mass trajectory. We further analyzed the timing performance of L_1 costs, showing that there is an increase in convergence time when using L_1 costs with Huber being the slowest, followed by SmoothL1 and Pseudo-Huber. Finally, we showed that the weight parameter λ has a similar effect for satellite control as it does on the cartpole—low values of λ lead to artifacts and high values of λ lead to high task costs. In the satellite problem, however, high λ did not numerically lead to more sparsity, as it produced longer thruster peaks for the lower control-limit thrusters, instead penalizing the high-limit thrusters more.

Finally, we note that it is important to enforce control limits when using sparsity-inducing losses. For unconstrained methods (DDP [2], FDDP [5]) clamping of applied controls in the forward-pass works in practice but results in slower convergence and often leads to getting stuck in local minima. We tested the losses with active-set control-limited DDP [18], and in particular BoxFDDP [19] in our experiments due to its greater generalization without an initial guess.

Future directions for this research are in time optimization of trajectories using DDP and to address the convergence and control artifacts using regularization strategies.

REFERENCES

[1] D. Malyuta, T. Reynolds, M. Szmuk, B. Acikmese, and M. Mesbahi, "Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints," in *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2020.

[2] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *Int. J. Control*, 1966.

[3] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control," in *IEEE/RSJ IROS*, Oct 2018, pp. 1–9.

[4] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, July 2017.

[5] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *IEEE ICRA*, May 2020, pp. 2536–2542.

[6] M. Kelly, "Trajectory Optimization," 2017. [Online]. Available: http://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/index.html

[7] H. Ferrolho, W. Merkt, V. Ivan, W. Wolfslag, and S. Vijayakumar, "Optimizing dynamic trajectories for robustness to disturbances using polytopic projections," in *IEEE/RSJ IROS*, 2020.

[8] E. N. Hartley, M. Gallieri, and J. M. Maciejowski, "Terminal spacecraft rendezvous and capture with LASSO model predictive control," *Int. J. Control*, vol. 86, no. 11, pp. 2104–2113, Nov. 2013.

[9] S. Le Cleac'h and Z. Manchester, "Fast Solution of Optimal Control Problems With L1 Cost," *AAS/AIAA Astrodynamics Specialist Conference*, vol. 904, pp. 1–11, 2019.

[10] E. M. Hoffman, M. P. Polverini, A. Laurenzi, and N. G. Tsagarakis, "A Study on Sparse Hierarchical Inverse Kinematics Algorithms for Humanoid Robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 235–242, Jan 2020.

[11] M. Schmidt, G. Fung, and R. Rosales, "Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches," in *Machine Learning: ECML 2007*, J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič, *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4701, pp. 286–297, iSSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science.

[12] P. J. Huber, *Robust Statistics*. John Wiley & Sons, 2004.

[13] C.-Y. Fu, M. Shvets, and A. C. Berg, "RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free," *arXiv:1901.03353 [cs]*, Jan. 2019, arXiv: 1901.03353 version: 1. [Online]. Available: http://arxiv.org/abs/1901.03353

[14] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996, publisher: [Royal Statistical Society, Wiley]. [Online]. Available: https://www.jstor.org/stable/2346178

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[16] D. Q. Mayne, "Differential Dynamic Programming–A Unified Approach to the Optimization of Dynamic Systems," in *Control and Dynamic Systems*, C. T. Leondes, Ed. Academic Press, Jan. 1973, vol. 10, pp. 179–254.

[17] V. Ivan, Y. Yang, W. Merkt, M. P. Camilleri, and S. Vijayakumar, *EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control*. Cham: Springer International Publishing, 2019, pp. 211–240. [Online]. Available: https://doi.org/10.1007/978-3-319-91590-6_7

[18] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE ICRA*, May 2014, pp. 1168–1175.

[19] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Sola, *et al.*, "A Direct-Indirect Hybridization Approach to Control-Limited DDP," 2020.