CACTO-SL: Using Sobolev Learning to improve Continuous Actor-Critic with Trajectory Optimization

Elisa Alboni[1]   ELISA.ALBONI@UNITN.IT

Gianluigi Grandesso[1]   GIANLUIGI.GRANDESSO@UNITN.IT

Gastone P. Rosati Papini[1]   GASTONE.ROSATIPAPINI@UNITN.IT

Justin Carpentier[2]   JUSTIN.CARPENTIER@INRIA.FR

Andrea Del Prete[1]   ANDREA.DELPRETE@UNITN.IT

[1] *Dept. of Industrial Engineering, University of Trento, Italy*
[2] *INRIA, Paris, France*

Abstract

Trajectory Optimization (TO) and Reinforcement Learning (RL) are powerful and complementary tools to solve optimal control problems. On the one hand, TO can efficiently compute locally-optimal solutions, but it tends to get stuck in local minima if the problem is not convex. On the other hand, RL is typically less sensitive to non-convexity, but it requires a much higher computational effort. Recently, we have proposed CACTO (Continuous Actor-Critic with Trajectory Optimization), an algorithm that uses TO to guide the exploration of an actor-critic RL algorithm. In turns, the policy encoded by the actor is used to warm-start TO, closing the loop between TO and RL. In this work, we present CACTO-SL, an extension of CACTO exploiting the idea of Sobolev Learning. To make the training of the critic network faster and more data efficient, we enrich it with the gradient of the Value function, computed via a backward pass of the differential dynamic programming algorithm. Our results show that the new algorithm is more efficient than the original CACTO, reducing the number of TO episodes by a factor ranging from 3 to 10, and consequently the computation time. Moreover, we show that CACTO-SL helps TO to find better minima and to produce more consistent results.

Keywords: Trajectory Optimization, Reinforcement Learning, Sobolev Learning, Global Optimization

1. Introduction

Robot control challenges have long been addressed through Trajectory Optimization (TO). The high-level desired task is encoded in the cost function of an Optimal Control Problem (OCP), which is minimised with respect to the OCP decision variables, which represent the state and control trajectories. Constraints are added to ensure that the solution of the OCP takes into account the robot dynamics, the actuator bounds, and the task-related constraints. When tackling complex problems, the OCP may feature a highly non-convex cost and/or highly nonlinear dynamics. Therefore, gradient-based solvers frequently encounter local minima and are unable to find a globally optimal solution. There exist TO methods based on the Hamilton-Jacobi-Bellman equation or Dynamic Programming, for continuous-time and discrete-time problems, respectively, that can find a globally optimal solution. However, these methods are hindered by the curse of dimensionality, which limits their applicability.

ALBONI[1] GRANDESSO[1] ROSATI PAPINI[1] CARPENTIER[2] DEL PRETE[1]

With the emergence of deep Reinforcement Learning (RL) and its application to the continuous domain, this machine learning tool is applied more and more widely to robot control problems showing impressive results on continuous state and control spaces Lillicrap et al. (2015); Fujimoto et al. (2018); Haarnoja et al. (2018); Schulman et al. (2017). RL algorithms are less prone to converge to local minima due to their exploratory nature. Yet, there are still several challenges related to the application of RL to robot control, such as the necessity for extensive exploration.

As a potential solution to overcome the limitations of RL and TO, we have recently presented the CACTO algorithm (Continuous Actor-Critic with Trajectory Optimization) Grandesso et al. (2023). CACTO iteratively leverages the explorative nature of RL to initialize TO to escape local minima, while exploiting TO to guide the RL exploration. Thanks to the interplay of TO and RL, CACTO's policy provides TO with initial guesses that allow it to obtain better trajectories than with other initialization techniques. While the use of TO demonstrated to efficiently accelerate convergence of RL, the computational burden associated with solving TO episodes has posed some limitations. In particular, as the system complexity increases, this issue becomes more relevant, hindering the scalability of the algorithm. The ability of TO to produce the Value function's gradient with limited computational cost presents an opportunity to enhance the algorithm's data efficiency, by increasing the information extracted from each TO problem. By leveraging Sobolev Learning (SL), the gradient information can be incorporated in the critic's training, enhancing the performance of the algorithm.

Sobolev spaces are metric spaces where the distance between functions is defined in terms of both the difference between the function values and the difference between their derivatives values. The universal approximation theorem for neural networks in Sobolev spaces Hornik (1991) shows that, under some assumptions, not only a neural network can approximate the value of a function, but also its derivatives with respect to its inputs. This work inspired other research, such as Czarnecki et al. (2017), which extensively studied the employment of the neural network derivatives to improve the training process in different kinds of problems, including policy distillation and regression on datasets. Sobolev Learning found applications also in the robot control field. For example, in Parag et al. (2022), this technique is used to learn a Value function to be used as an approximate terminal cost in an OCP, thus allowing to shorten the problem horizon, and speeding up the solver. In Le Lidec et al. (2023), stochastic Sobolev Learning is used to include computationally efficient higher-order information in the policy training, resulting in improved sample efficiency and stability. Even though the use of the derivatives comes with a computational overhead, Sobolev Learning increases robustness against noise and improves generalization as well as data-efficiency Masuoka (1993); Lee and Oh (1997). In this work, we incorporate Sobolev Learning in CACTO to improve its efficiency and scalability. Our main contributions are:

- We present a new version of the CACTO algorithm that computes the gradient of the Value function using the backward pass of the differential dynamic programming algorithm, and uses it to improve the training of the critic network.

- We show that using ReLU activation functions is detrimental when training the critic with Sobolev Learning, and we address this issue by switching to smooth periodic activation functions.

- We integrated our open-source implementation of CACTO with the software libraries *CasADi* Andersson et al. (2019) (for numerical optimization) and *Pinocchio* Carpentier et al. (2019) (for multi-body dynamics), making it more versatile and easily accessible by the community.

- We exemplify the behavior of our algorithm in the companion video using a simple 1D toy problem.

2. Method

In Grandesso et al. (2023), we presented CACTO, an algorithm for finding globally-optimal control policies through TO-guided actor-critic RL, which we summarize in Section 2.1. To increase its computational efficiency, we propose to use Sobolev Learning Czarnecki et al. (2017) for training the critic network, as detailed in Section 2.2.

2.1. Original CACTO Formulation

This section presents the former formulation of CACTO, an optimization algorithm designed to address discrete-time optimal control problems with a finite time horizon, with the following structure:

$$\underset{X,U}{\text{minimize}} \quad L(X,U) = \sum_{t=0}^{T-1} l_t(x_t, u_t) + l_T(x_T) \tag{1a}$$

$$\text{subject to} \quad x_{t+1} = f(x_t, u_t) \quad \forall t = 0 \ldots T-1, \tag{1b}$$

$$|u_t| \leq u_{max} \quad \forall t = 0 \ldots T-1, \tag{1c}$$

$$x_0 = x_{init} \tag{1d}$$

where the decision variables are the state and control sequences denoted as $X = x_{0 \ldots T}$ and $U = u_{0 \ldots T-1}$, with $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$. The cost function $L(\cdot)$ is defined as the sum of the running costs $l_t(x_t, u_t)$ and the terminal cost $l_T(x_T)$. The dynamics, control limits and initial conditions are represented by (1b), (1c) and (1d).

The algorithm begins with the TO-phase, where it solves N TO problems (1) with random initial states x_{init}, random time horizons in $[1, T]$, and using a classic warm-starting technique, e.g. initializing states to x_{init} and control inputs to zero. For each optimal state computed by TO, we compute the partial L-step cost-to-go, where L is the number of lookahead steps used for Temporal Difference (TD) learning, and store them in a replay buffer along with the relative transition (i.e., state, control, and state after L steps). The states stored in the replay buffer are augmented states $\tilde{x} = [x, t]$, where t is the time. Next we start the update phase: for M times, a batch of N_B transitions is sampled from the replay buffer and used to update the neural networks of the critic and the actor. In particular, the critic loss is the mean squared error between the critic's output and the so-called TD target, while the actor loss is the action-value (i.e., Q) function. Finally, a rollout of the actor's policy is used to warm-start the TO problems in the next episodes, closing the loop. In Grandesso et al. (2023), we showed the capabilities of CACTO to escape local minima, while being more computationally efficient than state-of-the-art RL algorithms. Moreover, CACTO has been proven to converge to a global optimum in a discrete-space setting.

2.2. CACTO with Sobolev Learning (CACTO-SL)

To exploit Sobolev Learning we need to provide the gradient of the Value function with respect to the state: V_x. To analytically compute V_x, we use the backward-pass of *Differential Dynamic Programming* (DDP) Jacobson (1968); Tassa et al. (2012), an efficient optimal control method for
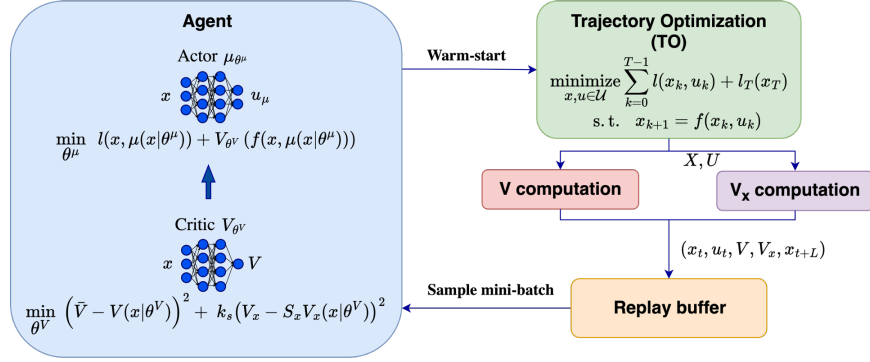
Figure 1: Scheme of the CACTO-SL algorithm.

unconstrained nonlinear problems. A scheme of the CACTO-SL algorithm is depicted in Fig. 1. Notice that, since DDP does not handle control bounds, the OCPs solved with CACTO-SL do not feature control bounds, but penalties.

2.2.1. USING THE GRADIENT OF THE VALUE FUNCTION IN CACTO-SL

To exploit V_x when training the critic, we store it in the replay buffer along with the associated transition. Then, during the update-phase, the critic parameters θ^V are updated to match both the target values $\bar{V}$ and their gradients V_x:

$$\underset{\theta^V}{\text{minimize}} \quad \frac{1}{N_B} \sum_{i=1}^{N_B} \left(\bar{V}_i - V(\tilde{x}_i | \theta^V) \right)^2 + k_S \left(V_{x,i} - S_x V_{\tilde{x}}(\tilde{x}_i | \theta^V) \right)^2, \tag{2}$$

where N_B is the batch size and the matrix $S_x = \begin{bmatrix} I_{n \times n} & 0_n \end{bmatrix} \in \mathbb{R}^{n \times n+1}$ selects the first n elements of the vector $V_{\tilde{x}}(\tilde{x}_i | \theta^V) \in \mathbb{R}^{n+1}$, thus excluding from the error the derivative of V w.r.t. t. This is because the state in CACTO is augmented with the time t, therefore the gradient of the critic contains also the derivative w.r.t. t. However, DDP does not compute it because it uses a discrete representation for the time.

The relative weight of the two components in (2) is defined by the coefficient k_S. Tuning k_S to achieve the right trade-off is important for optimal performance. As k_S gets smaller, CACTO-SL tends to the original CACTO algorithm. Setting k_S to a large value may seem sensible because the training of the actor, which is our ultimate goal, relies solely on V_x. However, we should consider that the Value function is likely discontinuous at the boundaries of the basins of attraction of the different locally-optimal solutions of the OCP. Around discontinuities, the gradient fails to capture the local behavior of the function, therefore setting k_S too large may result in extremely poor approximations of the Value around discontinuities. This could slow down, or even prevent, convergence to a globally-optimal policy.

To achieve an ideal trade-off, given the rapid changes and extended range of values taken by V_x in our problems, we introduce a symmetric logarithmic function in the gradient-related loss:

$$\underset{\theta^V}{\text{minimize}} \quad \frac{1}{N_B} \sum_{i=1}^{N_B} \left(\bar{V}_i - V(\tilde{x}_i | \theta^V) \right)^2 + k_S \left(\text{logsym}(V_{x,i}) - \text{logsym}(S_x V_{\tilde{x}}(\tilde{x}_i | \theta^V)) \right)^2, \tag{3}$$

4

where the function logsym(x) is defined as

$$\text{logsym}(x) = \begin{cases} \log(x+1) & \text{if } x \geq 0 \\ -\log(-x+1) & \text{if } x < 0 \end{cases} \tag{4}$$

The logsym function reduces the relative importance of potential large (either positive or negative) values in the gradient, which may arise when the network tries to approximate a discontinuous Value function (see the companion video for a visual illustration of this phenomenon).

2.2.2. DIFFERENTIABILITY OF THE CRITIC NETWORK

The original CACTO algorithm used ReLU activation functions for both actor and critic. In CACTO-SL, we have observed that ReLU functions do not work well with Sobolev Learning. Indeed, ReLU functions lead to piecewise-linear approximations, for which the gradient may not represent well the local behavior of the function. For this reason, in CACTO-SL the critic network uses SIREN (Sinusoidal Representation Networks) layers Sitzmann et al. (2020). Characterized by its smooth and continuously differentiable nature, SIREN layers bring a crucial advantage to learning the gradient, preventing the generation of ill-behaved gradients from the loss term on the derivative mismatch.

2.2.3. SAMPLE EFFICIENCY

By incorporating the Value gradient, the amount of information provided by each transition is much higher than in CACTO: each transition contains both the partial cost-to-go (1 scalar) and the gradient of the Value function (an *n*-dimensional vector). Therefore, while ensuring sufficient exploration of the state space is still necessary, we can expect fewer TO episodes to be needed for learning a good approximation of the Value function. This was empirically confirmed by the fact that we could benefit from increasing the ratio between the number of network updates and the number of TO episodes. In particular, with the aim of reducing the total number of TO episodes, we increased both the number of TO episodes before each update phase, e_{update}, and the number of updates performed at each algorithm iteration, K. We start with a small value of K, because it is not useful to accurately learn the initial policy, which represents the TO solver. As the algorithm progresses, we increase K, performing more network updates after each TO-phase, as the accumulated data are expected to represent the Value function of a policy closer and closer to global optimality.

This strategic adjustment enables a significant reduction in the number of TO episodes (by a factor of 3 to 10) and, as a consequence, in the computation time.

2.2.4. SOFTWARE FOR TRAJECTORY OPTIMIZATION AND MULTI-BODY DYNAMICS

While CACTO relied on *Pyomo* Nicholson et al. (2018) for solving TO problems, CACTO-SL switched to *CasADi* Andersson et al. (2019), an open-source Automatic Differentiation framework for numerical optimization. By leveraging its symbolic framework and automatic differentiation capabilities, CasADi enables efficient computation of the cost function's derivatives, required for implementing Sobolev Learning. Moreover, *CasADi* is compatible with *Pinocchio* Carpentier et al. (2019), a versatile rigid-body dynamics library, which freed us from the burden of hand-coding the system dynamics. Each TO problem is transcribed using collocation, and then solved with the nonlinear optimization solver *IpOpt* Wächter and Biegler (2006). Finally, to speed up the code, we parallelized the generation of the warm-start trajectories, the TO problems, and the computation of the cost-to-go and its gradient.

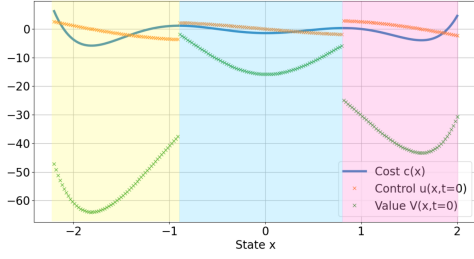ALBONI[1] GRANDESSO[1] ROSATI PAPINI[1] CARPENTIER[2] DEL PRETE[1]

Figure 2: Cost, control and Value obtained using a naive initial guess.
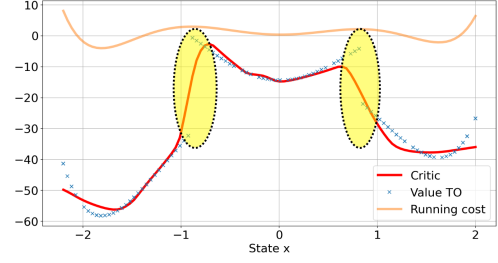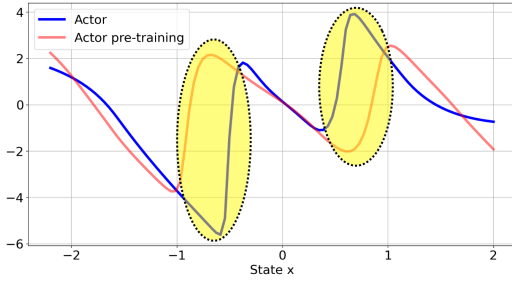

Figure 3: Critic smooths Value's discontinuities.

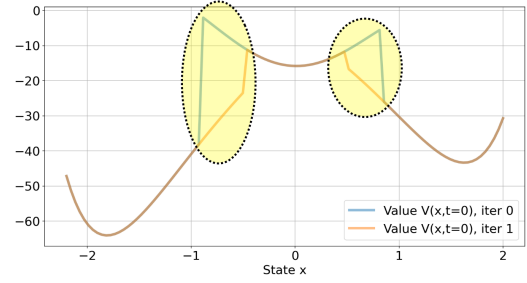Figure 4: Actor after first iteration.


Figure 5: Value after first iteration.

3. Results

3.1. 1-Dimensional Toy Problem

This section exemplifies the behaviour of CACTO and the effect of Sobolev Learning using a 1D toy problem, with a 1D state, a single integrator dynamics and a cost function with three local minima (Fig. 2). Solving TO problems with a naive initial guess, highlights the presence of three basins of attraction (see Fig. 2). Notice that $V(x)$ is not continuous at the boundaries of these basins. We now analyse the first iteration of CACTO. First, we solve 100 TO problems and train the critic network. Fig. 3 shows that the network smooths out the discontinuities because the critic network cannot represent discontinuous functions. After pre-training the actor network to imitate the TO control inputs, the actor network is trained to minimize the Q function. Fig. 4 and Fig. 5 show that the resulting actor provides a better policy, as the basin of attraction of the worst minimum is shrunk and the Value obtained by initializing TO with the trained actor policy has improved.

We now analyse the effect of Sobolev Learning. As mentioned above, the critic is trained to predict the Value resulting from TO, which is typically discontinuous at the boundaries of the basins of attraction associated to the different local minima. This leads to a poor fit of the gradient in proximity of the discontinuities, and at the boundaries of the explored space. Introducing Sobolev Learning helps to smooth out the discontinuities because the critic tries to match also V_x, see Fig. 6 and Fig. 7. As shown in Fig. 6 and Fig. 7, increasing k_S, the smoothing increases, which can help the actor to improve faster. However, k_S should not be increased too much because it may lead to a complete removal of the discontinuities. This alters the real Value of the states and may prevent convergence to a globally optimal policy.

Interested readers can find a more detailed explanation in the companion video.

6
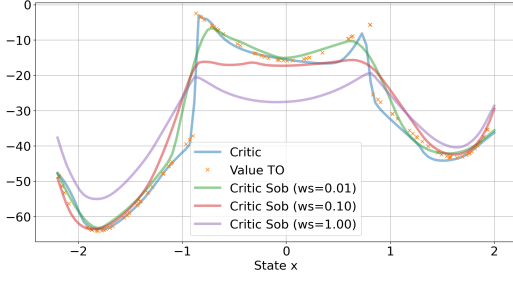
Figure 6: Effect of k_s on critic learning: V.
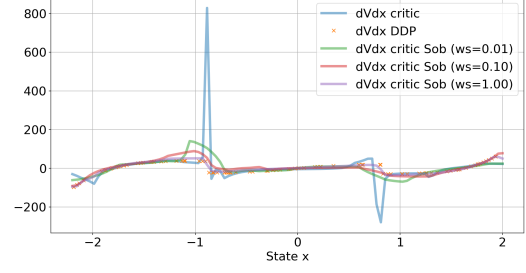
Figure 7: Effect of k_s on critic learning: V_x.

3.2. Empirical Results

This section presents our empirical results, with the goal to understand whether the use of Sobolev-Learning in CACTO is actually beneficial. The ability of CACTO-SL to provide good warm-starts to TO problems is analysed in the same scenarios presented in Grandesso et al. (2023). The source code is available on the GitHub page of the project.

The analysed task consists in reaching in the shortest time a desired final position, while avoiding three ellipse-shaped obstacles, and minimizing the control effort. The task is described by the following running cost:

$$
l(x,u) = \underbrace{w_d||p_{ee} - p_g||^2}_{l_1(x)} + \underbrace{-\frac{w_p}{\alpha_1} \ln(e^{-\alpha_1\left(\sqrt{(x_{ee}-x_g)^2+c_2}+\sqrt{(y_{ee}-y_g)^2+c_3}+c_4\right)} + 1)}_{l_2(x)} +
$$
$$
+ \underbrace{\frac{w_{ob}}{\alpha_2}\sum_{i=1}^{3} \ln\left(e^{-\alpha_2\left(\frac{(x_{ee}-x_{ob,i})^2}{(a_i/2)^2} + \frac{(y_{ee}-y_{ob,i})^2}{(b_i/2)^2} - 1\right)} + 1\right)}_{l_3(x)} + \underbrace{w_u||u||_2^2}_{l_4(u)} + \underbrace{\left\|\frac{u}{u_{max}}\right\|_2^{10}}_{l_5(u)}
$$

(5)

where l_1 penalizes the distance between $p_{ee} = (x_{ee}, y_{ee})$ (the x-y coordinate of the system's end-effector) and p_g (the goal position to be reached); l_2 encodes a cost valley in the neighborhood of the goal position; l_3 penalizes collision with the three obstacles centered in $p_{ob,i} = (x_{ob,i}, y_{ob,i})$ with axes a_i, and b_i; the w_*'s are user-defined weights; c_2, c_3, c_4, α_1, and α_2 are the parameters of the softmax functions; l_4 and l_5 are the control regularization and penalty to ensure that the optimal controls are in the desired range. The terminal cost is equal to the running cost, except for l_4 and l_5. Fig. 8 depicts the cost function, neglecting the control-effort term and control-penalty.

The task is designed to ensure the presence of many local minima. In particular, if the system starts from the *Hard Region*, highlighted by a green rectangle in Fig. 8, it is very hard for the solver to provide a globally optimal solution. In all our tests, we compute the critic targets with TD(50) (i.e. $L = 50$), which empirically led to a stable training of critic and actor.

3.3. Comparison between activation functions

We compared the critic network based on SIREN layers with the critic networks based on ReLU and based on ELU, another smooth activation layer, in combination with Sobolev Learning using $k_S = 10^3$. Fig. 9 shows that using the SIREN layers (or also ELU layers, but to a slightly lower extent) leads to a critic that clearly captures the lower values of states inside the C-shaped obstacle.
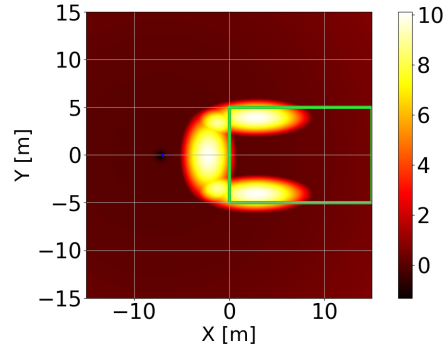
Figure 8: Cost function (5) without the control effort term, considering a target at $p_g = [-7, 0]$ with weights $w_d = 10^{-3}$, $w_p = 5$ and $w_{ob} = 10^1$. The green rectangle delimits the *Hard Region*.
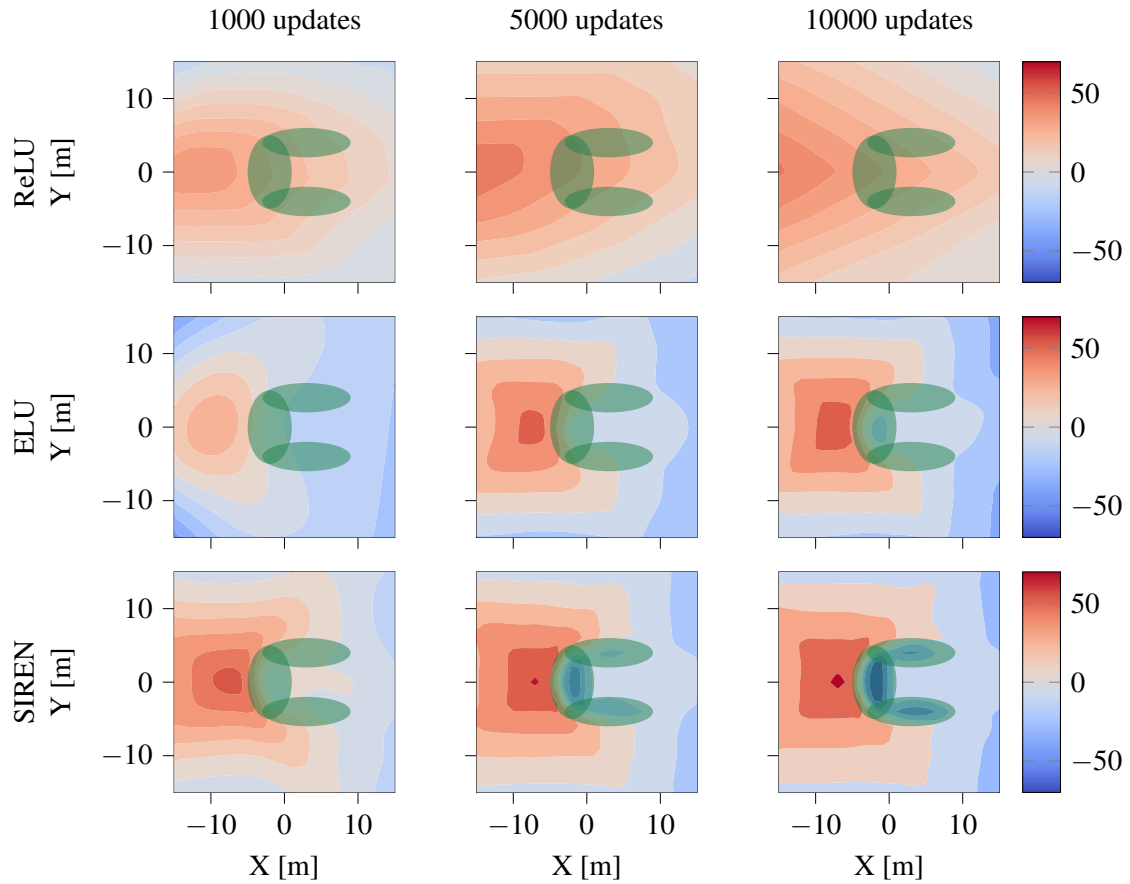

Figure 9: Comparison of critic networks based on ReLU (top), ELU (middle) and sinusoidal (bottom) activation functions at 1000, 5000 and 10000 updates for a point mass with single integrator dynamics.

On the other hand, this is completely overlooked by the critic using ReLU functions, even after 10000 iterations.

3.4. Comparison between CACTO-SL and CACTO

The comparison between CACTO-SL and CACTO is conducted with a 2D double integrator, and a Dubins car, using $k_s = 10^3$. We compared the algorithms by computing the mean cost of the trajectories obtained by initializing TO with the policy's rollout. We used as initial states a grid of points over the analysed XY space, with mesh size equal to 1 m. The analysis focuses on the *Hard Region* ($x \in [0, 15]$ m and $y \in [-5, 5]$ m). For an easier interpretation of the results, the initial velocity of the system is set to zero. In the Dubins car test, the initial heading is randomly chosen. For each test, we performed 5 runs using the same seeds for both CACTO and CACTO-SL to minimize the effect of the network initialization.

3.4.1. DOUBLE-INTEGRATOR SYSTEM

We consider a 2D double integrator, with state vector $[x, y, v_x, v_y, t] \in \mathbb{R}^5$ and control vector $[a_x, a_y] \in \mathbb{R}^2$. The maximum episode length is 10 s. The target point is $p_g = [-7, 0]$. In contrast to the previous version of CACTO, where we alternated $e_{update} = 25$ TO episodes and $K = 80$ network updates, in CACTO-SL we perform $e_{update} = 200$ TO episodes and an increasing number of network updates, starting from $K = 1000$ up to $K = 15000$. Fig. 10 reports the median (across 5 runs) of the mean cost (across initial conditions) starting from the *Hard Region* against the number of TO episodes obtained initializing the TO problems with CACTO and CACTO-SL. In both cases, we stopped the training after 50k updates, when both algorithms have converged to a stable TO solution.

Fig. 10 shows that CACTO-SL has led to a 92% reduction in the number of TO problems. As regards the computation time, it is important to recall that it heavily depends on the algorithm implementation, on the hardware, and on the number of cores used in the parallelization. Parallelizing the execution on 10 cores, the decrease of TO problems has resulted in a reduction of just 7% of the total execution time because the system is quite simple. However, without parallelization, the total computation time is reduced from 111 to 49 minutes. Moreover, Fig. 10 shows a significant reduction in the variance across the five runs. This outcome can be attributed to the enhanced training of the critic through Sobolev Learning. The improved training results in a more consistent set of critic networks. As a consequence, the actors converge towards similar policies, leading to a more uniform initial guess across runs.

3.4.2. CAR SYSTEM

The second problem involves a Dubins car modelled as a point mass. The 6D state vector consists in the displacement along x and y, angular displacement, linear velocity, linear acceleration and time. The 2D control vector contains the angular velocity and the linear jerk. The car should reach the target point $p_g = [-7, 0]$ m. No penalty is added on its orientation. We updated the networks every $e_{episode} = 500$ TO episodes, using an increasing number of updates, starting from $K = 1000$ and reaching $K = 15000$. In CACTO instead the ratio between TO episodes and network updates was 25/160. Fig. 10 shows the results. In this case, 170k updates have been performed.

In this case, CACTO-SL leads to a notable reduction in the number of TO problems solved and, consequently, in the computation time, from 10 h to 6.30 h (parallelizing the TO problems on 10 cores). Note that the reduction in computation time increases significantly also with parallelization,

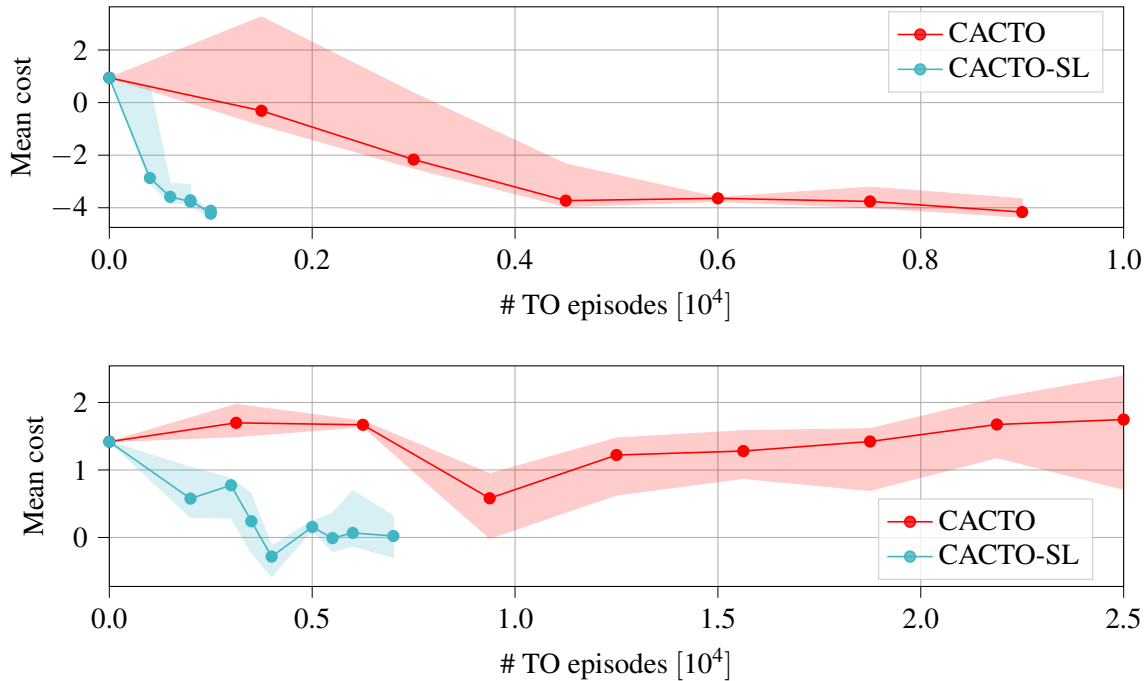ALBONI[1] GRANDESSO[1] ROSATI PAPINI[1] CARPENTIER[2] DEL PRETE[1]

Figure 10: Median (across 5 runs) of the mean cost (across initial conditions) starting from the *Hard Region* with zero initial velocity against the number of TO episodes for a point mass with double integrator dynamics (top) and a Dubins car (bottom). The shaded area represents the area between the first and third quartiles.

even though the Dubins car has only one more state than the 2D double integrator. Also in this test, Sobolev Learning plays an important role in achieving smaller variance across the runs.

4. Conclusions

This paper presented CACTO-SL, an extension of the CACTO algorithm, featuring Sobolev Learning. CACTO-SL efficiently computes the gradient of the Value function using the backward pass of the DDP algorithm, and exploits this information to improve the training of the critic network. Our results show that this makes CACTO-SL significantly more sample efficient than CACTO, reducing the number of TO episodes by a factor ranging from 3 to 10. This also leads to almost halving the computation time required to reach a stable solution. Moreover, our results show that CACTO-SL reduces the variance of the algorithm across different runs, and helps TO to find better minima.

In the future, we plan to switch to a BoxDDP backward pass Tassa et al. (2014), which would allow us to consider hard bounds on the control inputs. Moreover, we are exploring different approaches to bias the sampling of initial states towards more informative state regions, which should improve even further the sample efficiency of CACTO and allow us to scale better to more complex systems.

Acknowledgments

References

Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.

Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.

Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Gianluigi Grandesso, Elisa Alboni, Gastone P. Rosati Papini, Patrick M. Wensing, and Andrea Del Prete. Cacto: Continuous actor-critic with trajectory optimization—towards global optimality. *IEEE Robotics and Automation Letters*, 8(6):3318–3325, 2023. doi: 10.1109/LRA.2023. 3266985.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

David H Jacobson. New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach. *Journal of Optimization Theory and Applications*, 2:411–440, 1968.

Quentin Le Lidec, Wilson Jallet, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Enforcing the consensus between trajectory optimization and policy learning for precise robot control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 946–952. IEEE, 2023.

Jeong-Woo Lee and Jun-Ho Oh. Hybrid learning of mapping and its jacobian in multilayer neural networks. *Neural computation*, 9(5):937–958, 1997.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Ryusuke Masuoka. Noise robustness of ebnn learning. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pages 1665–1668. IEEE, 1993.

Bethany Nicholson, John D. Siirola, Jean Paul Watson, Victor M. Zavala, and Lorenz T. Biegler. Pyomo.Dae: a Modeling and Automatic Discretization Framework for Optimization With Differential and Algebraic Equations. *Mathematical Programming Computation*, 10(2):187–223, 2018. ISSN 18672957. doi: 10.1007/s12532-017-0127-0. URL https://doi.org/10.1007/s12532-017-0127-0.

Amit Parag, Sébastien Kleff, Léo Saci, Nicolas Mansard, and Olivier Stasse. Value learning from trajectory optimization and sobolev descent: A step toward reinforcement learning with superlinear convergence properties. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 01–07, 2022. doi: 10.1109/ICRA46639.2022.9811993.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 4906–4913, 2012. ISBN 9781467317351. URL https://dada.cs.washington.edu/homes/todorov/papers/MPCGetUp.pdf.

Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1168–1175, 2014. ISBN 9781479936854. doi: 10.1109/ICRA.2014.6907001.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.