2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Equality Constrained Differential Dynamic Programming

Sarah El Kazdadi*, Justin Carpentier* and Jean Ponce*

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561339

***Abstract*— Trajectory optimization is an important tool in task-based robot motion planning, due to its generality and convergence guarantees under some mild conditions. It is often used as a post-processing operation to smooth out trajectories that are generated by probabilistic methods or to directly control the robot motion. Unconstrained trajectory optimization problems have been well studied, and are commonly solved using Differential Dynamic Programming methods that allow for fast convergence at a relatively low computational cost. In this paper, we propose an augmented Lagrangian approach that extends these ideas to equality-constrained trajectory optimization problems, while maintaining a balance between convergence speed and numerical stability. We illustrate our contributions on various standard robotic problems and highlights their benefits compared to standard approaches.**

I. Introduction

Optimal control and trajectory optimization in general are powerful and versatile mathematical frameworks to program robot's motions. They provide a way to efficiently compute, in a generic manner, complex robot motions which minimize a given cost criterion and satisfy a set of constraints, while taking advantage of the dynamical capacities of the robot in question. The most well-known optimal control principles [1], i.e., the Pontryagin Maximum Principle (PMP) and Hamilton-Jacobi-Bellman (HJB) equations, have been derived at the same time in the 50s. For a certain period, *indirect* approaches [2], which directly exploit these principles, were preferred to analytically solve optimal control problems, but could only be applied to a limited class of dynamical systems (e.g space rockets, Dubin's car). With the growth of the computational resources available on standard computers, together with the development of efficient numerical optimization methods for solving large scale problems, *direct* approaches [2] have emerged as an efficient alternative for solving optimal control problems. These methods [2], such as single shooting, multiple-shooting or collocations, first, discretize the problem, reformulate it as a constrained nonlinear programming (NLP) instance, and then solve it using classic methods for constrained optimization [3] such as Penalty, Sequential Quadratic Programming (SQP), Interior Points (IP) or Augmented Lagrangian (AL) methods. While these approaches can handle a large variety of dynamical systems [4], [5], [6], their performances directly rely on the efficiency and numerical robustness of the underlying NLP solvers and their ability to exploit the problem sparsity.

More recently, Differential Dynamic Programming (DDP) originally introduced in the late 60s [7], has emerged as a good trade-off between direct and indirect approaches. DDP belongs to the class of second-order methods (similar to Newton methods), and is relatively simple to implement while taking advantage of the sparsity pattern of the problem. Additionally, it also provides, along with the solution, a linear feedback term that can be used to correct the control sequence when the observed trajectory deviates from the optimal one. In particular, this allows the solution to be robust to some amount of external noise. Classical DDP was initially limited to unconstrained problems. Recent works have proposed using various optimization strategies to handle different levels of constraints: box control constraints (aka box-DDP) [8] via box Quadratic Programming (QP), nonlinear equality constraints [9], [10], [11], [12], a mix of box constraints and equality constraints, and generic nonlinear constraints [13], [14], [15] via either AL, penalty, or SQP methods. However, these solutions either provide limited convergence guarantees (no globalization strategies [3]) or require a significant amount of iterations of the DDP algorithm or the internal numerical routines to converge to a feasible solution, limiting then the deployment of these methods for online Model Predictive Control for instance. Other works

In this paper, we extend the DDP method to handle additional equality constraints, beyond the constraints of the dynamical system itself, using an augmented Lagrangian-based formulation. The first contribution of this paper is an adaptation of the globalization strategy called Bound-Constrained Lagrangian (BCL) [16], [3] to the case of equality-constrained DDP. This goes beyond previous works which only provide limited convergence guarantees to find an optimal solution. As the second contribution, we propose two slightly different strategies to estimate the Lagrange multipliers of the constraints: one that directly applies the BCL method to DDP to converge to an optimal solution, and a second one that more closely mirrors the principles of the DDP method, by also providing a feedback term, extending the solution to a neighborhood of the optimal trajectory.

The paper is organized as follows. We first recall the main technical background in Sec. II concerning DDP and AL. Sec. III depicts the main contributions of the paper and Sec. IV empirically shows the performances of the proposed approach on several standard robotic problems.

II. Differential Dynamic Programming and Augmented Lagrangian

In this section, we recall the main equations behind the unconstrained DDP and the Augmented Lagrangian methods. This section also introduces the main notations used throughout the paper.

*The authors are with Inria and Département d'Informatique de l'Ecole Normale Suprieure, PSL Research University, Paris, France.
Corresponding author: sarah.el-kazdadi@inria.fr

A. Optimal control

A discrete-time dynamical system is a system whose evolution from the time instant i to $i+1$ is governed by a transition function f_i:

$$x_{i+1} = f_i(x_i, u_i), \tag{1}$$

where $x \in \mathbf{X}$, $u \in \mathbf{U}$, and $\mathbf{X}$ (resp. $\mathbf{U}$) is the state space (resp. control space) of the dynamical system. A feasible trajectory $(\mathbf{x}, \mathbf{u})$ is a sequence of states $\mathbf{x} = (x_0, \dots, x_N)$ and controls $\mathbf{u} = (u_0, \dots, u_{N-1})$ that satisfies (1).

A trajectory optimization problem is defined by a dynamical system f_i, a running cost, $\ell_i : \mathbf{X} \times \mathbf{U} \to \mathbb{R}$, as well as a terminal cost $\ell_{\mathrm{f}} : \mathbf{X} \to \mathbb{R}$. The total cost of a control sequence $\mathbf{u}$, starting from an initial state x_0 corresponds to:

$$J(x_0, \mathbf{u}) = \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_{\mathrm{f}}(x_N), \tag{2}$$

where the state sequence is obtained by integrating the dynamics of the system along the time horizon N. The goal is to find the optimal control sequence that minimizes the total cost, given a fixed initial state:

$$\mathbf{u}^\star = \operatorname*{argmin}_{\mathbf{u} \in \mathbf{U}^N} J(x_0, \mathbf{u}). \tag{3}$$

B. Unconstrained differential dynamic programming

A DDP problem is solved by iteratively improving a starting feasible trajectory also known as the initial guess. Let $\mathbf{u_i} = (u_i, \dots, u_{N-1})$ be the tail of the control sequence starting at the time i. We define the *cost-to-go* functions as:

$$J_i(x, \mathbf{u_i}) = \sum_{j=i}^{N-1} \ell_i(x_i, u_i) + \ell_{\mathrm{f}}(x_N)$$

and the value functions as:

$$V_i(x) = \min_{\mathbf{u_i} \in \mathbf{U}^{N-i}} J_i(x, \mathbf{u_i}).$$

This optimality condition on the sequence of $(V_i)_{i \in \{0,\dots,N\}}$ translates into the Bellman equation:

$$V_i(x) = \min_{u \in \mathbf{U}} \ \ell_i(x,u) + V_{i+1}(f_i(x,u)) \text{ with } V_N(x) = \ell_{\mathrm{f}}(x). \tag{4}$$

We also define the Q function as the argument of the minimization operator:

$$Q_i(x,u) = \ell_i(x,u) + V_{i+1}(f_i(x,u)). \tag{5}$$

Then, to find the optimal control policy at time i, one needs to solve:

$$\operatorname*{argmin}_{u \in \mathbf{U}} Q_i(x,u). \tag{6}$$

As (4) or (6) may not have an analytical solution in general, they are modeled with second-order approximations. The idea behind DDP is to compute a second-order approximation of V_i, propagating them backward in time using the Bellman recursion. The computed controls minimizing the previous expression (6) are then used as a refinement of the controls from the current trajectory.

Backward pass — Propagating the second-order approximation: We model the functions V_i and Q_i using second-order approximations, and compute their Jacobians and Hessians recursively, starting from

$$V_i(x + \mathrm{d}x) = v + V_x \mathrm{d}x + \frac{1}{2} \mathrm{d}x^\top V_{xx} \mathrm{d}x.$$

where v, V_x and $V_x x$ are respectively the value, the Jacobian, and the Hessian at x.

And for each $i \in N-1, \dots, 0$, we define the second-order approximation of Q_i around (x_i, u_i) as:

$$\begin{aligned} Q_i(x + \mathrm{d}x, u + \mathrm{d}u) = q + Q_x \mathrm{d}x + Q_u \mathrm{d}u \\ + \frac{1}{2} \mathrm{d}x^\top Q_{xx} \mathrm{d}x + \frac{1}{2} \mathrm{d}u^\top Q_{uu} \mathrm{d}u + \mathrm{d}u^\top Q_{ux} \mathrm{d}x. \end{aligned}$$

The equalities (4) then read:

$$\text{For } i = N : V_{xx} = \ell_{\mathrm{f},xx},\ V_x = \ell_{\mathrm{f},x}, \text{ and } v = \ell_{\mathrm{f}}. \tag{8}$$

For $i \in \{N-1, \dots, 0\}$, we denote V_{i+1} by V'.

$$Q_{xx} = \ell_{xx} + f_x^\top V'_{xx} f_x + V_x'^\top f_{xx}, \tag{9a}$$

$$Q_{ux} = \ell_{ux} + f_u^\top V'_{xx} f_x + V_x'^\top f_{ux}, \tag{9b}$$

$$Q_{uu} = \ell_{uu} + f_u^\top V'_{xx} f_u + V_x'^\top f_{uu}, \tag{9c}$$

$$Q_x = \ell_x + V_x'^\top f_x, \tag{9d}$$

$$Q_u = \ell_u + V_x'^\top f_u, \tag{9e}$$

$$q = \ell + v'. \tag{9f}$$

The minimizer of (6) is then found by taking a Newton step, and can be written as:

$$u^\star(x) = k + K \mathrm{d}x, \text{ with } k = -Q_{uu}^{-1} Q_u \text{ and } K = -Q_{uu}^{-1} Q_{ux}. \tag{10}$$

Note that when the current solution is not close enough to the optimal solution, Q_{uu} may not necessarily be positive definite. In this case a regularization term λI may be added.

Finally, the second-order approximation of the value function at the time instant i is given by:

$$V_{xx} = Q_{xx} - K^\top Q_{uu} K, \tag{11a}$$

$$V_x = Q_x - k^\top Q_{uu} K, \tag{11b}$$

$$v = q - \frac{1}{2} k^\top Q_{uu} k. \tag{11c}$$

Forward pass — Integrating the system dynamics: The refined control policies are given by k_i, K_i. To obtain an improved trajectory, we integrate the system using these new policies. A line search parameter α is used to guarantee convergence, as the full step may not necessarily decrease the total cost function (2) unless we are close to the optimum. The new trajectory is then given by:

$$u_i^\star = u_i + \alpha k + K(x_i^\star - x_i), \tag{12a}$$

$$x_0^\star = x_0 \text{ and } x_{i+1}^\star = f_i(x_i^\star, u_i^\star). \tag{12b}$$

C. The Augmented Lagrangian

AL [17] is a standard method to solve nonlinear constrained optimization problems. We first review the AL approach for solving equality-constrained QPs before presenting the algorithm for solving generic nonlinear equality-constrained problems with global convergence guarantees.

Equality-constrained Quadratic Programming: Consider the following equality constrained quadratic program:

$$\min_x \ \frac{1}{2}x^\top Qx + q^\top x, \tag{13}$$

$$\text{s.t.} \quad Ax = b, \tag{14}$$

where Q is symmetric and A is full rank. The augmented Lagrangian of this QP is:

$$\mathscr{L}_\mu(x,\lambda) = \frac{1}{2}x^\top Qx + q^\top x + \lambda^\top(Ax-b) + \frac{\mu}{2}\|Ax-b\|^2. \tag{15}$$

The classic AL method consists of minimizing $\mathscr{L}_\mu$ with respect to x. In this case, the minimizer is given by:

$$x' = -(Q+\mu A^\top A)^{-1}(q + A^\top\lambda - \mu A^\top b) \tag{16}$$

with the updated multipliers:

$$\lambda' = \lambda + \mu(Ax' - b). \tag{17}$$

Each minimization + update iteration decreases the error on the constraint by a factor that grows proportionally with the penalty term μ as it tends to ∞. Also, note that the augmented Lagrangian can only be minimized if $Q+\mu A^\top A$ is positive definite, which constrains μ to be larger than some lower bound. Because of this, it is often preferable to use an optimization strategy that increases the value of μ if the system cannot be solved or the convergence rate is below the desired threshold as explained below.

When μ becomes too large, the condition number of $(Q+\mu A^\top A)$ blows up which may lead to unstable computations. To overcome this issue, solving (16) and (17) is strictly equivalent to solving the system:

$$\begin{bmatrix} Q & A^\top \\ A & -\frac{1}{\mu}I \end{bmatrix}\begin{bmatrix} x' \\ \lambda' \end{bmatrix} = \begin{bmatrix} -q \\ b - \frac{\lambda}{\mu} \end{bmatrix} \tag{18}$$

with the main advantage of having to invert a matrix with a better condition number.

Nonlinear equality constrained optimization: In general, the constrained optimization problems we aim to solve in robotics go beyond QPs and are generically formulated as:

$$\min_x \ f(x) \ \text{s.t.} \ c(x) = 0, \tag{19}$$

where f is a smooth twice-differentiable cost function and c is a smooth equality constraint. The augmented Lagrangian function now reads:

$$\mathscr{L}_\mu(x,\lambda) = f(x) + \lambda^\top c(x) + \frac{\mu}{2}\|c(x)\|^2. \tag{20}$$

Within this generic setting, additional care must be taken in order to converge to the solution when using AL methods. A common strategy relies on applying a gradient descent or Newton step with a backtracking line search to minimize the augmented Lagrangian $\mathscr{L}_\mu$ with respect to x, then updating the multipliers with the standard update rule: $\lambda' = \lambda + \mu c(x)$. The penalty factor μ is increased after each iteration to ensure that the problem is convex in a neighborhood of the solution. In practice, however, this approach often fails to converge to the solution when the step-size does not decrease rapidly enough. A regular increase of μ can also make it more difficult to properly minimize the augmented Lagrangian with a line search approach, when the $\frac{\mu}{2}\|c(x)\|^2$ term starts to dominate the others. On top of that, the high-penalty parameter values interact poorly with the DDP algorithm, as the high-order terms can propagate during the backward pass, which affects the stability of the numerical computations.

We propose instead an alternative called Bound-Constrained Lagrangian [16], [3] and detailed in Alg. 1. It consists of only updating the multipliers when both the optimum of the augmented Lagrangian is reached, which we detect based on the norm of the gradient, and when the error of the constraints is small enough. If the optimum of the augmented Lagrangian is reached, but the constraint has not decreased sufficiently, we then increase the value of μ by a constant factor k. The BCL strategy is the basis of the NLP solver LANCELOT [16] and comes with general convergence guarantees under mild assumptions, often fulfilled in robotics. As highlighted in Sec. IV and shown in [16], this strategy allows to (i) convergence to a locally optimal solution from any starting conditions, and (ii) to significantly reduce the number of required steps, by enforcing a consensus between primal and dual updates. We propose to adapt this strategy for handling constraints in DDP. A similar strategy was used in [18], with varying degrees of success depending on the accuracy of the computed derivatives.

Algorithm 1: Bound-Constrained Lagrangian [16]

Result: Optimal primal and dual variables $x^\star$ and $\lambda^\star$
Given an initial x_0, λ_0, and parameters η_0, ω_0, μ_0;
Given hyperparameters $k > 1, \alpha \in (0,1)$;
while $\eta_n > \eta_{threshold}$ *and* $\omega_n > \omega_{threshold}$ **do**
  $x_{n+1} = x_n - t\nabla^2_{x,x}\mathscr{L}_\mu{}^{-1}\nabla_x\mathscr{L}_\mu$ with $t > 0$ found by line search;
  if $\|\nabla_x\mathscr{L}_\mu(x_{n+1},\lambda_n)\|_\infty < \omega_n$ **then**
    if $\|c(x_{n+1})\|_\infty < \eta_n$ **then**
      $\lambda_{n+1} = \lambda_n + \mu_n c(x_{n+1})$;
      $\mu_{n+1} = \mu_n$; $\eta_{n+1} = \eta_n/\mu_n^\alpha$; $\omega_{n+1} = \omega_n/\mu_n$;
    else
      $\mu_{n+1} = k\mu_n$;
    end
  end
end

Imposing a desired convergence rate: We also propose an alternative strategy to update the penalty factor μ to control the convergence rate. This strategy consists of increasing the penalty parameter adaptively, instead of using a constant factor k. Asymptotically, the constraint error $\|c(x_n)\|$ between two consecutive multiplier updates decreases at a rate of $O(\mu_n^{-1})$. We can estimate the decrease factor ϕ by keeping track of the norm constraint error at the previous optimum and computing the ratio of the new value. $c'_0 = \|c(x_0)\|$, and $c'_{n+1} = \{\|c_(x_{n+1})\|$ if λ_{n+1} is updated, otherwise $c'_n\}$. ϕ is then estimated by letting $c'_{n+1} = \frac{\phi}{\mu_n}c'_n$. Then, assuming we

want a convergence rate of $1/\mu_{n+1}^{\beta}$, with $\beta \in (0,1)$, we can deduce the ideal value of μ_{n+1} by letting $\phi/\mu_{n+1} = 1/\mu_{n+1}^{\beta}$. The solution is given by $\mu_{n+1} = \phi^{\frac{1}{1-\beta}}$.

III. Equality Constrained Differential Dynamic Programming

This section contains the core contribution of the paper. We consider the optimal control problem, characterized by the dynamics (1) and the cost function (2), under the additional equality constraints over x_i, u_i, given by:

$$h_i(x_i, u_i) = 0, \tag{21}$$

where $h_i : \mathbf{X} \times \mathbf{U} \to \mathbb{R}^{c_i}$, and c_i is lower than or equal to the dimension of the tangent space of $\mathbf{U}$. In order to solve the optimization problem, we suggest two approaches based on the well-known augmented Lagrangian methods.

A. Globally constant Lagrange multipliers

In this case, the augmented Lagrangian of the constrained DDP is given by:

$$\mathscr{L}_\mu(\mathbf{x}, \mathbf{u}, \lambda) = \sum_{i=0}^{N-1} \left\{ \ell_i(x_i, u_i) + \lambda_i^\top h_i(x_i, u_i) + \frac{\mu}{2} \|h_i(x_i, u_i)\|^2 \right\} + \ell_f(x_T) \tag{22}$$

The first strategy consists of (i) fixing the value of the multipliers, then (ii) finding the trajectory that minimizes the augmented Lagrangian, by relying on the previous unconstrained DDP algorithm. The forward and backward passes of the standard DDP algorithm remain largely unchanged. Eqs (9) have to be augmented with the AL terms:

$$\begin{aligned}
Q_{xx}^c &= Q_{xx} + (\lambda + \mu h)^\top h_{xx} + \mu h_x^\top h_x, \\
Q_{ux}^c &= Q_{ux} + (\lambda + \mu h)^\top h_{ux} + \mu h_u^\top h_x, \\
Q_{uu}^c &= Q_{uu} + (\lambda + \mu h)^\top h_{uu} + \mu h_u^\top h_u, \\
Q_x^c &= Q_x + (\lambda + \mu h)^\top h_x, \;\; Q_u^c = Q_u + (\lambda + \mu h)^\top h_u, \\
q^c &= q + \lambda^\top h + \frac{\mu}{2} \|h\|^2.
\end{aligned}$$

The DDP iterations are repeated until the minimizer of (22) is found with respect to $(\mathbf{x}, \mathbf{u})$. We can then update the value of the multipliers using the classic update:

$$\lambda_i' = \lambda_i + \mu h_i(x_i', u_i'), \tag{24}$$

To overcome the issue of ill-conditioning when inverting Q_{uu}^c, we use the same strategy as the one exposed in (18).

Note that similarly to the generic augmented Lagrangian method, additional care must be taken in practice to ensure convergence, by increasing μ when needed (e.g., when Q_{uu} is not positive definite, when solving the DDP subproblem does not bring us closer to satisfying the constraints, etc.), by following the BCL strategy recalled in Alg. 1. As this method is a direct adaptation of the classic augmented Lagrangian procedure for solving generic optimization problems, we can readily borrow the convergence results to prove the desired properties of our algorithm. That is, if we converge to a feasible solution, then it is optimal, and the convergence is linear with a rate of approximately $\frac{1}{\mu}$. Importantly, it is also possible to let $\mu \to \infty$ when we detect that we are close enough to the optimum and the constraints are linearly independent. This is equivalent to using second-order sequential quadratic programming methods, and can allow for quadratic convergence.

Unfortunately, the control policy given by the DDP iterations no longer holds the same meaning. In the unconstrained case, the feedback term allows correcting the control when the actual trajectory differs from the expected state. But in our current scenario, even correcting back to a feasible trajectory is no longer feasible, once the constraints are violated. Our second approach was designed to fix this issue.

B. Locally affine Lagrange multipliers

The following method is similar to the previous one, with the difference that the multipliers are now allowed to vary with the state of the system ($\lambda_i : \mathbf{X} \to \mathbb{R}^c$). We choose to represent them as affine functions defined in a neighborhood of x_i, which take the form

$$\lambda_i(x) = \lambda_i + \lambda_{i,x}(x - x_i). \tag{25}$$

In this case, the augmented Lagrangian is equal to:

$$\mathscr{L}_\mu(\mathbf{x}, \mathbf{u}, \lambda) = \sum_{i=0}^{N-1} \left\{ \ell_i(x_i, u_i) + \lambda_i(x_i)^\top h_i(x_i, u_i) + \frac{\mu}{2} \|h_i(x_i, u_i)\|^2 \right\} + \ell_f(x_T). \tag{26}$$

The equations of the backward pass (9) incorporate new terms:

$$\begin{aligned}
Q_{xx}^c &= Q_{xx} + (\lambda + \mu h)^\top h_{xx} + \mu h_x^\top h_x + h_x^\top \lambda_x + \lambda_x^\top h_x, \\
Q_{ux}^c &= Q_{ux} + (\lambda + \mu h)^\top h_{ux} + \mu h_u^\top h_x + h_u^\top \lambda_x, \\
Q_{uu}^c &= Q_{uu} + (\lambda + \mu h)^\top h_{uu} + \mu h_u^\top h_u, \\
Q_x^c &= Q_x + (\lambda + \mu h)^\top h_x + h^\top \lambda_x, \;\; Q_u^c = Q_u + (\lambda + \mu h)^\top h_u, \\
q^c &= q + \lambda^\top h + \frac{\mu}{2} \|h\|^2.
\end{aligned}$$

The same considerations of the first method still apply, with regards to increasing μ when necessary, to ensure convergence. In addition to that, since the multipliers are computed in a neighborhood of the trajectory's states x_i, the origin of the approximation and the value at the origin are updated at the end of the forward pass of each DDP iteration. When $\mathbf{X}$ is a vector space, this produces the same affine function. So the Jacobian λ_x is unchanged and the value at the origin is updated as $\lambda' = \lambda + \lambda_x(x' - x)$. Once the minimizer of the augmented Lagrangian is found, the multipliers can be updated at each node i using the constraint feasibility error:

$$\lambda' = \lambda + \mu h \text{ and } \lambda_x' = \lambda_x + \mu(h_x + h_u K). \tag{28}$$

This method converges at a similar rate to the first one, with the additional benefit that the computed feedback allows us to recompute the optimal control that satisfies the equality constraints, in a neighborhood of the solution trajectory. The pseudo-code of the algorithm is depicted in Algo. 2.

One limitation, however, is that convergence can only be guaranteed when the optimal policy exists, which adds the

constraint that h_u and $h_{ux,i}$ must have full rank. Otherwise, changing the control would have no effect (at first order) on the feasibility of the constraints, making it infeasible to find a policy that always satisfies them. An important consequence of this is that constraints of the form $h(x) = 0$ cannot be directly handled by this method, as h_u, h_{ux} are always zero. It is possible to get around this limitation in certain cases. A constraint $h_i(x_i)$ can also be written as $h_i(f(x_{i-1}, u_{i-1}))$, which creates a dependence on u_{i-1}. This means that for $i < T$ we can choose to redefine the constraint as:

$$h'_i(x_i, u_i) = h_{i+1}(f(x_i, u_i), u_i). \tag{29}$$

This discards the constraint $h_0 = 0$, but this is unavoidable in the general case.

Algorithm 2: BCL-DDP

Result: Optimal trajectory $\mathbf{u}^\star, \mathbf{x}^\star, \lambda^\star, \lambda_x^\star$
Given an initial $\mathbf{u}_0, \lambda_0, \lambda_x$, and parameters η_0, ω_0, μ_0;
Given hyperparameters $k > 1, \alpha \in (0,1)$;
while $\eta_n > \eta_{threshold}$ *and* $\omega_n > \omega_{threshold}$ **do**
  Backward pass (27) to get feedback k_{n+1}, K_{n+1};
  Forward pass (12) with LS;
  $\mathbf{u}_{n+1}, \mathbf{x}_{n+1}$ that decreases $\mathscr{L}_\mu$;
  if $\|\nabla_u \mathscr{L}_\mu(\mathbf{x}_{n+1}, \mathbf{u}_{n+1}, \lambda_n(\mathbf{x}_{n+1}))\|_\infty < \omega_n$ **then**
    if $\|c(\mathbf{x}_{n+1})\|_\infty < \eta_n$ **then**
      λ_{n+1} = AL multiplier update (28);
      $\mu_{n+1} = \mu_n$; $\eta_{n+1} = \eta_n / \mu_n^\alpha$; $\omega_{n+1} = \omega_n / \mu_n$;
    else
      $\mu_{n+1} = k\mu_n$ or any other update rule;
    end
  end
end

C. Handling infeasible initial guesses

In some cases where the evolution of the system may be unstable (e.g., due to unstable dynamics or a large temporal horizon), it may be desirable to use a multiple shooting [19] method instead, as they allow for infeasible trajectories which can be easier to keep under control. In this case, we can readily modify our method to allow for a discontinuous trajectory. Given a dynamical system over the state space $\mathbf{X}$, the control space $\mathbf{U}$ where the evolution is dictated by the transition function f_i, and additional constraints h_i, we can transform it into the system over the same state space, with controls in $u \in \mathbf{U}, \varepsilon \in T\mathbf{X}_{x_{i+1}}$, the tangent space of $\mathbf{X}$ at x_{i+1}, such that the dynamics are:

$$x_{i+1} = f'_i(x_i, u_i, \varepsilon_i) = f(x_i, u_i) + \varepsilon_i, \tag{30}$$

under the constraints:

$$h_i(x_i, u_i) = 0 \text{ and } a\varepsilon_i = 0, \tag{31a}$$

where $a > 0$ is a parameter controlling how the infeasibility penalty is weighted *w.r.t.* the standard constraints. Since convergence to the constraint zero set happens progressively along with the minimization of the objective function, rather than eagerly as in SQP methods, this allows the second constraint $\varepsilon_i = 0$ to control the degree of infeasibility, and then improving the overall stability of the optimization process.

D. Related works

The closest work to our paper is the ALTRO algorithm [13]. It is also based on an AL strategy and composed of two main stages: the first stage corresponds to an augmented Lagrangian strategy applied to DDP, similar to Sec III-A, except that the multipliers are always updated and the penalty factor constantly increased. As illustrated in IV and detailed in the literature [3], [16], this may lead to poor convergence results (convergence to a feasible but sub-optimal solution), especially when working with finite precision. The second stage of ALTRO consists of a projection step, whose capacities to converge to a good optimum actively depend on the first stage, which does not come with convergence guarantees. Compared to this work, relying on BCL as a globalization strategy enables us to converge toward local optima with a controlled convergence rate. Other approaches have used a Penalty based method [15] without a guarantee of the convergence to an optimal solution. The SQP-based strategy proposed by [14] requires solving multiple QPs when computing a feasible solution for the forward pass, which might slow down the computations. In our work instead, equality-constrained QPs are solved in the backward pass, not by calling an external QP solver, but directly when implementing the AL approach, as highlighted by (18). In addition, our work is the first to introduce a linear dependency on the state quantity to the evolution of the multipliers, so far improving the feedback term of the constrained DDP solver.

IV. Results

In order to validate and analyze the theoretical and practical properties of the methods introduced in III, we have developed an open-source C++ multi-precision-arithmetic (MA) framework[1] to solve equality constrained DDP problems. This framework relies on Eigen [20] and GNU MPFR [21] libraries for linear algebra and MA. It is based on Pinocchio [22], [23] for computing the robot dynamics and their analytical derivatives [24]. In practice, we performed the experiments using either double floating-point precision ($\approx 10^{-16}$ relative error) or a much higher precision ($\approx 10^{-500}$ relative error).

A. Cartpole System

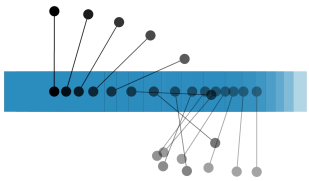

Fig. 1: Cartpole: optimal trajectory found by our constrained DDP solver.

[1] https://github.com/s-elkazdadi/ddp-pinocchio

The first experiment (augmented precision) showcases a cartpole system, where the control variable is the force applied to the cart. The objective and constraint functions were defined such that the upright position must be reached, while minimizing the total norm of the applied force.

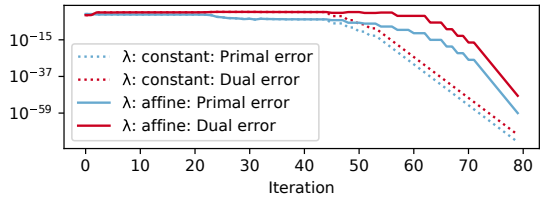

Fig. 2: Convergence rate: μ is increased by a constant factor of 100

We compare the two penalty update strategies, Fig. 2 uses

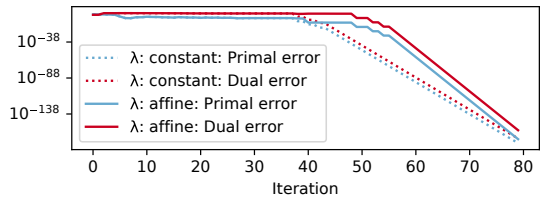

Fig. 3: Convergence rate: μ is increased adaptively

the strategy described in Alg. 1, and increases the penalty parameter whenever the minimum of the augmented Lagrangian is reached without a sufficient decrease of the constraint error. The algorithm used in Fig. 3 uses an adaptive update strategy by using the shrinking rate of the constraint error to estimate the optimal penalty parameter. In order to avoid increasing the penalty too excessively early on in the solving procedure, we limit the growth factor by a certain amount. ($\mu_{n+1} < 10^6 \mu_n$). Both methods converge to the correct solution, but the second one manages to find an appropriate penalty parameter in fewer iterations

Note that in general, the multiplier and penalty update strategy are important in order to guarantee the convergence. Otherwise, we may fail to converge to the optimal value or diverge to infinity, as can be seen in Fig.4.

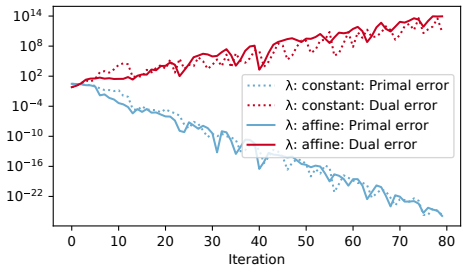

Fig. 4: Convergence rate: every iteration, λ is updated and μ is increased by a factor of 3

We can see that the algorithm variant that builds an affine model of the multipliers takes longer to converge in most cases. This is since, on top of finding the optimal trajectory, the feedback term needs to be computed in parallel as well, which can help with stability when the trajectory needs to be corrected in real-time. Asymptotically, when a noise of norm ε is added to the trajectory during the rollout, the order of the error with the constant multiplier variant is $O(1/\varepsilon\mu)$, while the order of the variant with affine multipliers is $O(1/\varepsilon^2)$.

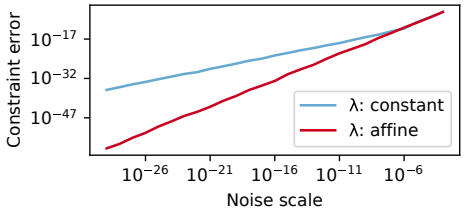

Fig. 5: Evolution of the constraint error as a function of the noisiness of the dynamics

B. Robotic arm: UR5

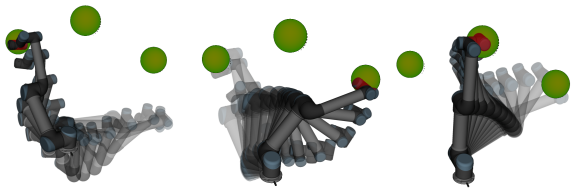

Fig. 6: Optimal trajectory of the UR-5 robot reaching the three targets.

The second experiment (usual precision) is performed on a robotic arm equipped with 6 degrees of freedom. The system dynamics are integrated over 2 seconds, with a discretization step of $dt = 0.01s$. Once again, the objective function minimizes the cost, but the constraint was set so that the robot must reach the given targets sequentially at three consecutive time instants. Despite the instability of the system, the algorithm manages to converge to within an error of 10^{-6} of the optimal solution in approximately 50 iterations.

V. Discussion and Conclusion

The proposed augmented Lagrangian approach to deal with equality constrained DDP displays consistent convergence properties, despite the genericity of the algorithm. This allows the DDP methods to be reliably applied to a large class of optimal control problems, where constraints can be used to generate more robust and feasible movements. However, there is still room for improvement to better take into account real-world application requirements. Most notably, inequality constraints are a natural next step, seeing as classical augmented Lagrangian methods already tend to handle them naturally. They play a key role in guaranteeing the robustness of robot motion planning, as they can express the boundaries of the valid controls, states, and the limits that are induced by contact and friction forces.

REFERENCES

[1] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[2] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 65–93.

[3] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[4] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[5] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1398–1404.

[6] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[7] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[8] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[9] D. M. Murray and S. J. Yakowitz, "Constrained differential dynamic programming and its application to multireservoir control," *Water Resources Research*, vol. 15, no. 5, pp. 1017–1027, 1979.

[10] M. Giftthaler and J. Buchli, "A projection approach to equality constrained iterative linear quadratic optimal control," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 61–66.

[11] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[12] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2017.7989016

[13] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7674–7679.

[14] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 695–702.

[15] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[16] A. R. Conn, G. Gould, and P. L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. Springer Science & Business Media, 2013, vol. 17.

[17] M. J. Powell, "A method for nonlinear constraints in minimization problems," *Optimization*, pp. 283–298, 1969.

[18] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," 09 2017, pp. 5674–5680.

[19] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[20] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[21] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "Mpfr: A multiple-precision binary floating-point library with correct rounding," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 2, pp. 13–es, 2007.

[22] J. Carpentier, F. Valenza, N. Mansard *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2019.

[23] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. 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.

[24] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems*, 2018.