**proxddp: a primal-dual solver for constrained trajectory optimization**

Journées Nationales de la Robotique Humanoïde, 2022

Wilson Jallet\textsuperscript{a,b,*}, Antoine Bambade\textsuperscript{b,c}, Nicolas Mansard\textsuperscript{a} and Justin Carpentier\textsuperscript{b}

**Abstract**— Trajectory optimization are problems are a fundamental component of solving optimal control problems (OCPs) on high-dimensional, complex robotic systems. It relies on two key components: 1/ the transcription into a sparse nonlinear program, and 2/ an appropriate algorithm which iteratively computes a solution within a desired convergence threshold. On one hand, differential dynamic programming (DDP) \cite{1} is an efficient framework for transcription and resolution of OCPs as a finite-dimensional nonlinear program, which fully exploits the sparsity structure induced by time. On the other hand, augmented Lagrangian methods \cite{2}-\cite{4} provide a framework for efficient algorithms that allow advanced constraint-satisfaction strategies. In the sequel, we discuss our work on building efficient algorithms for constrained trajectory optimization, based on DDP and augmented Lagrangian methods, as well as our upcoming C++ package for modelling and solving such problems. The features of our methods include: targeting problems with both equality and inequality constraints, being able to solve them within good accuracy, and being able to handle multiple-shooting formulations naturally (including the use of advanced implicit integrators e.g. variational integrators \cite{5}).

I. INTRODUCTION

In recent work \cite{6}-\cite{8}, new variants of DDP were introduced; firstly for equality-constrained problems \cite{6}, \cite{7} and then with multiple-shooting capabilities for use with implicit integrators (e.g. variational \cite{5}) \cite{7}, based on augmented Lagrangians. Then, further extensions to our framework were made to include inequality-constrained problems \cite{8}, based on a novel primal-dual augmented Lagrangian formulation for constrained programming which has successfully been used in quadratic programming \cite{9}.

\textit{a) Problem setting:} A typical trajectory optimization problem in continuous time can be formulated as the following infinite-dimensional program:

\[
\min_{x,u} \int_0^T \ell(t, x(t), u(t)) \, dt + \ell_1(x(T))
\]

\[\text{s.t. } \dot{x}(t) = f(x(t), u(t)), \quad x(0) = \bar{x}_0 \]

\[h(t, x(t), u(t)) \leq 0 \text{ (path constraints)}
\]

\[h_T(x(T)) \leq 0
\]

\[\text{b) Transcription:} \quad \text{In} \ [7], \text{we argue that for stiff systems (involving contact dynamics, for instance), it is desirable to get increased numerical stability in integration rules, which can be achieved by leveraging implicit numerical integrators (such as the midpoint rule, the class of implicit Runge-Kutta methods, or variational integrators like \cite{5}). In general, assuming the discrete sequence of states and controls are } \{x_k\}_{k=0}^{N-1}, \{u_k\}_{k=0}^{N-1} \text{ respectively, an implicit integration rule can be written in the form } f^d(x_k, u_k, x_{k+1}) = 0 \text{ where the superscript "d" means "discretised" (it will dropped in the sequel). The equation above can be solved (using Newton-Raphson for instance) to obtain the next state } x_{k+1} \text{ from } (x_k, u_k). \text{ Some authors such as } [10] \text{ solve for } x_{k+1} \text{ and apply the implicit function theorem to obtain the derivatives of the mapping } (x_k, u_k) \mapsto x_{k+1}, \text{within a standard single-shooting framework. In our approach, outlined below, we go for a multiple-shooting formulation where dynamical feasibility is only asked for at convergence of the algorithm. The corresponding transcription of the OCP } [1], \text{ including the path constraints, reads:}
\]

\[
\min_{x,u} J(x,u) = \sum_{k=0}^{N-1} \ell_k(x_k, u_k) + \ell_1(x_N)
\]

\[\text{s.t. } f^d(x_k, u_k, x_{k+1}) = 0, \quad x_0 = \bar{x}_0 \quad (2b)
\]

\[h_k(x_k, u_k) \leq 0
\]

\[h_N(x_N) \leq 0.
\]

II. PROPOSED METHOD

\textit{a) Backward sweep and Bellman equation:} Our main idea is to consider the dynamic programming principle or Bellman equation for \cite{2}, which reads

\[
V_i(x) = \min_{u,y} \ell(x, u) + V_{i+1}(y)
\]

\[f(x, u, y) = 0, \quad h(x, u) \leq 0
\]

with boundary condition \(V_N(x) = \ell_1(x)\). In the sequel, we introduce the \(Q\)-function associated with the Lagrangian of \cite{5}, where \((\lambda, \nu)\) are the dual variables:

\[
Q(x,u,y,\lambda,\nu) = \ell(x,u) + V_{i+1}(y) + \lambda^T f(x,u,y) + \nu^T h(x,u).
\]

The Bellman equation then simply becomes the min-max problem

\[
V_i(x) = \min_{u,y} \max_{\lambda,\nu \geq 0} Q(x,u,y,\lambda,\nu).
\]
This Bellman equation can be relaxed by using the classical augmented Lagrangian method or the primal-dual augmented Lagrangian method of Gill & Robinson [11], extended to inequality constraints in [8] and [9]. This relaxation can be seen as a proximal-point iteration on the dual problem to (4), or in the dual variables in (5), as studied in [4].

We start with the (primal) augmented Lagrangian method, that of Powell, Hestenes and Rockafellar [4]. At step \( k \) of the overall algorithm, the Bellman iteration now depends on external multiplier estimates \( \lambda_k, \nu_k \) for the constraints,

\[
\min_{u,y} \ell(x, u) + V_{i+1}(y) + \frac{1}{2\mu} \left\| f(x, u, y) + \mu \lambda_k \right\|^2 \text{ for the constraints,}
\]

The minimand is the augmented Lagrangian \( \mathcal{L}_k(u, y; x, \mu) \) associated with the Bellman equation (3) which depends on the current state \( x \). This Bellman equation, just like in standard DDP, is handled by a single (quasi)-Newton step – by the dynamic programming principle, this backward sweep recovers a Newton step over the state-control trajectory \((x, u)\), with respect to the augmented Lagrangian of the initial problem (2).

For the primal-dual augmented Lagrangian, we consider a primal-dual step which includes both \((u, y)\) and multipliers \((\lambda, \nu)\).

\[
\min_{u,y,\lambda,\nu} \mathcal{L}_k(u, y; x, \mu) + \frac{1}{2\mu} \left\| f(x, u, y) + \mu (\lambda_k - \lambda) \right\|^2 \text{.}
\]

b) The KKT system and feedback gains: To compute the Newton or quasi-Newton step \((\delta u, \delta y, \delta \lambda, \delta \nu)\) in the primal and dual variables in either (6) or (7), the following system can be solved (using e.g. an indefinite Cholesky factorization):

\[
\begin{bmatrix}
\delta u \\
\delta y \\
\delta \lambda \\
\delta \nu
\end{bmatrix} = -
\begin{bmatrix}
Q_u + Q_{ux} \delta x \\
Q_y + Q_{yx} \delta x \\
[ h + h_{x,y} \delta x + \mu \lambda ]_+ - \mu \nu
\end{bmatrix},
\]

where the matrix \( K_\mu \) is the “KKT matrix” of the backward sweep is

\[
K_\mu \overset{\text{def}}{=} \begin{bmatrix}
Q_{uu} & Q_{uy} & f_y^T & S_A h_u^+ \\
Q_{yu} & Q_{yy} & f_y^T & S_A h_y^+ \\
f_u & f_y & -\mu I & S_A h_u \\
S_A h_u & S_A h_y & -\mu S_A
\end{bmatrix},
\]

and \( S_A \) is simply a selection matrix for the active set \( A \) of constraints in the augmented Lagrangian method (the indices \( j \) such as \([ h(x, u) + \mu \lambda ]_+ \geq 0\), see [9] for further details): \( S_A \) is diagonal with \([S_A]_{jj} = 1\) if the constraint is active \((j \in A)\), 0 otherwise.

III. EXPERIMENTAL SHOWCASE

The aim of our work is to go towards a full-fledged implementation of our method, named ProxDDP, in C++ for fast and efficient resolution of constrained trajectory optimization problems, as a successor to the Crocoddyl [12] software library. For this talk, we have included two experiments from previous works, figures [1] and [2], which show satisfactory behaviour of the solutions to some constrained problems.

