Control-Limited Differential Dynamic Programming

Yuval Tassa†, Nicolas Mansard* and Emo Todorov†

***Abstract*— Trajectory optimizers are a powerful class of methods for generating goal-directed robot motion. Differential Dynamic Programming (DDP) is an indirect method which optimizes only over the unconstrained control-space and is therefore fast enough to allow real-time control of a full humanoid robot on modern computers. Although indirect methods automatically take into account state constraints, control limits pose a difficulty. This is particularly problematic when an expensive robot is strong enough to break itself.**

In this paper, we demonstrate that simple heuristics used to enforce limits (clamping and penalizing) are not efficient in general. We then propose a generalization of DDP which accommodates box inequality constraints on the controls, without significantly sacrificing convergence quality or computational effort. We apply our algorithm to three simulated problems, including the 36-DoF HRP-2 robot. A movie of our results can be found here goo.gl/eeiMnn

I. Introduction

It would be appealing to specify the behavior of a robot in terms of simple cost functions, and let an intelligent control algorithm handle the details. This is also the idea behind the task-function [1] or the operational-space [2] approaches: instead of working in the configuration space, the motion is specified with a more abstract function related, for example, to the position of the end effector or to the output value of a sensor. The task-function approach naturally leads to inverse kinematics [3] or operational-space inverse dynamics [4] and is particularly active nowadays in humanoid robotics [5], [6], [7] where it is turned into control machinery by using task sequencing [7]. Classically, a simple proportional or proportional-derivative controller in the task space is used [8], but it results in simple trajectories that behave badly when coming close to obstacles or joint limits. The convergence basin of these local methods is then very small. Ad-hoc task trajectories can be learned [9], which enlarge the convergence basin with a-priori knowledge and provide a consistent way to define complex task trajectories, but this is difficult to generalize to new situations.

Trajectory optimization is the process of finding a state-control sequence which locally minimizes a given cost function. *Shooting methods* – which trace their ancestry to the two-point boundary-value problem of the venerable Maximum Principle [10] – are an important sub-class of trajectory optimization methods. Unlike so-called *direct methods* which explicitly represent the state, these methods parameterize only the controls, and obtain the states from forward integration (hence "shooting"). States are never explicitly represented in the optimization space and consequently these methods are also known as *indirect* [11]. Because the dynamics are folded into the optimization, state-control trajectories are always strictly feasible and "dynamic constraints" are unnecessary. If additionally the controls are unconstrained, so is the optimization search-space and shooting methods can enjoy the benefits of fully unconstrained optimization.

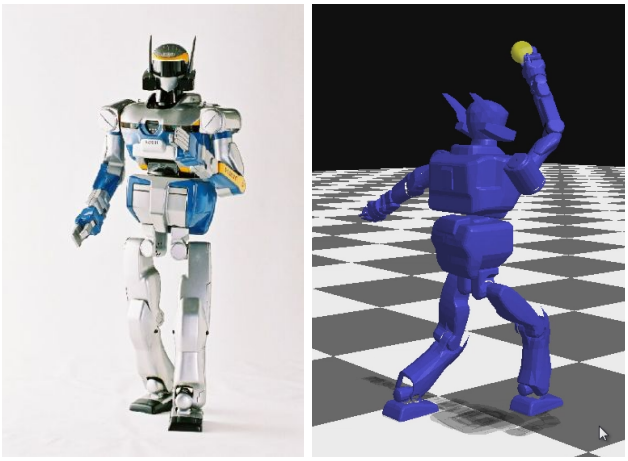

Fig. 1. ***Left:*** The humanoid robot HRP-2. ***Right:*** Real-time reaching and balancing behaviors are described in Section IV and in the attached movie.

DDP is a second-order shooting method [12] which under mild assumptions admits quadratic convergence for any system with smooth dynamics [13]. It has been shown to posses convergence properties similar, to or slightly better than, Newton's method performed on the entire control sequence [14]. Classic DDP requires second-order derivatives of the dynamics, which are usually the most expensive part of the computation. If only the first-order terms are kept, one obtains a Gauss-Newton approximation known as iterative-Linear-Quadratic Regulator (iLQR) [15], [16], which is similar to Riccati iterations, but accounts for the regularization and line-search required to handle the nonlinearity.

Work on constrained indirect methods began with [17], see [18] for a review. The work most closely related to ours is [19], see section III-C below. The reader may notice that these papers had been published several decades ago. More recent work on constrained trajectory optimization for robotics has mostly focused on direct methods [20][21][22]. In that context the problem is transcribed into a generic sequential quadratic programming (SQP) which easily admits both equality and inequality constraints. We suspect that the reason these methods have been more popular is the general availability of off-the-shelf optimization software for generic SQPs. Both types of methods display different characteristics

† Computer Science & Engineering, Univ. of Washington, Seattle, USA ({tassa,todorov}@cs.washington.edu).

* LAAS-CNRS, Univ. Toulouse, France (nmansard@laas.fr).

and tradeoffs. The direct approach discards the temporal structure and is forced to search in a constrained space which is slower, however it is far easier to find better optima through continuation. The indirect approach is faster and better suited for warm-starting, but is far more sensitive to local minima.

In this paper, we consider the solution of control-constrained problems using indirect methods. We show experimentally in simulation that simplistic ways of handling them are inefficient and detrimental to convergence. We then propose an original solution to explicitly take the inequalities into account using a projected-Newton QP solver which is in the general class of active-set methods. The capability of the method is shown in simulation on a wide range of systems (random linear systems, a nonholonomic car and a humanoid robot). In Section II, we quickly recall the Differential Dynamic Programming algorithm. We characterize the box-constrained control problem in Section III, along with the proposed original solution. Finally, Section IV describes the results, illustrating the usefulness of our approach.

II. Differential Dynamic Programming

This section recalls the basics of DDP that are necessary for the algorithm proposed in Section III. More details are available, see e.g. [12] for the historical presentation or [23] for a modern treatment using the same notations as below.

A. Local Dynamic Programming

We consider a system with discrete-time dynamics, but a similar derivation holds for the continuous case [12]. The dynamics is modeled by the generic function $\mathbf{f}$

$$\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i), \tag{1}$$

which describes the evolution from time i to i+1 of the state $\mathbf{x} \in \mathbb{R}^n$, given the control $\mathbf{u} \in \mathbb{R}^m$. A trajectory $\{\mathbf{X}, \mathbf{U}\}$ is a sequence of states $\mathbf{X} \triangleq \{\mathbf{x}_0, \mathbf{x}_1 \ldots, \mathbf{x}_N\}$, and corresponding controls $\mathbf{U} \triangleq \{\mathbf{u}_0, \mathbf{u}_1 \ldots, \mathbf{u}_{N-1}\}$ satisfying (1).

The *total cost* denoted by J is the sum of running costs ℓ and final cost ℓ_f, incurred when starting from $\mathbf{x}_0$ and applying $\mathbf{U}$ until the horizon N is reached:

$$J(\mathbf{x}_0, \mathbf{U}) = \sum_{i=0}^{N-1} \ell(\mathbf{x}_i, \mathbf{u}_i) + \ell_f(\mathbf{x}_N).$$

As discussed above, *indirect* methods represent the trajectory implicitly using only the controls $\mathbf{U}$. The states $\mathbf{X}$ are recovered by integration of (1) from the initial state $\mathbf{x}_0$. The solution of the optimal control problem is the minimizing control sequence

$$\mathbf{U}^* \triangleq \operatorname*{argmin}_{\mathbf{U}} J(\mathbf{x}_0, \mathbf{U}).$$

Letting $\mathbf{U}_i \triangleq \{\mathbf{u}_i, \mathbf{u}_{i+1} \ldots, \mathbf{u}_{N-1}\}$ be the tail of the control sequence, the *cost-to-go* J_i is the partial sum of costs from i to N:

$$J_i(\mathbf{x}, \mathbf{U}_i) = \sum_{j=i}^{N-1} \ell(\mathbf{x}_j, \mathbf{u}_j) + \ell_f(\mathbf{x}_N).$$

The *Value* at time i is the optimal cost-to-go starting at $\mathbf{x}$:

$$V_i(\mathbf{x}) \triangleq \min_{\mathbf{U}_i} J_i(\mathbf{x}, \mathbf{U}_i).$$

The Value of the final time is defined as $V_N(\mathbf{x}) \triangleq \ell_f(\mathbf{x}_N)$. The Dynamic Programming Principle then reduces the minimization over a sequence of controls $\mathbf{U}_i$, to a sequence of minimizations over a single control, proceeding backwards in time:

$$V(\mathbf{x}) = \min_{\mathbf{u}} [\ell(\mathbf{x}, \mathbf{u}) + V'(\mathbf{f}(\mathbf{x}, \mathbf{u}))] \tag{2}$$

In (2) and below we omit the time index i and use V' to denote the Value at the next time step.

B. Quadratic Approximation

DDP involves iterating a *forward pass* (or *rollout*) which integrates (1) for a given $\mathbf{U}$, followed by a *backward pass* which compute a local solution to (2) using a quadratic Taylor expansion. Let $Q(\delta\mathbf{x}, \delta\mathbf{u})$ be the change in the argument of the RHS of (2) as a function of small perturbations of the i-th nominal $(\mathbf{x}, \mathbf{u})$ pair:

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = \ell(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}) + V'(\mathbf{f}(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u})) \tag{3}$$

The Q-function is the discrete-time analogue of the Hamiltonian, sometimes known as the *pseudo-Hamiltonian*. The second-order expansion of Q is given by:

$$Q_{\mathbf{x}} = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V'_{\mathbf{x}} \tag{4a}$$
$$Q_{\mathbf{u}} = \ell_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V'_{\mathbf{x}} \tag{4b}$$
$$Q_{\mathbf{xx}} = \ell_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{xx}} \tag{4c}$$
$$Q_{\mathbf{ux}} = \ell_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{ux}} \tag{4d}$$
$$Q_{\mathbf{uu}} = \ell_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{u}} + V'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{uu}}. \tag{4e}$$

where the last terms of (4c, 4d, 4e) denote the product of a vector with a tensor. The optimal control modification $\delta\mathbf{u}^*$ for some state perturbation $\delta\mathbf{x}$, is obtained by minimizing the quadratic model:

$$\delta\mathbf{u}^*(\delta\mathbf{x}) = \operatorname*{argmin}_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = \mathbf{k} + \mathbf{K}\delta\mathbf{x}. \tag{5a}$$

This is a locally-linear feedback policy with

$$\mathbf{k} \triangleq -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} \qquad \text{and} \qquad \mathbf{K} \triangleq -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} \tag{5b}$$

the feed-forward modification and feedback gain matrix, respectively. Plugging this policy back into the expansion of Q, a quadratic model of V is obtained. After simplification it is

$$\Delta V = -\tfrac{1}{2} \mathbf{k}^{\mathsf{T}} Q_{\mathbf{uu}} \mathbf{k} \tag{6a}$$
$$V_{\mathbf{x}} = Q_{\mathbf{x}} - \mathbf{K}^{\mathsf{T}} Q_{\mathbf{uu}} \mathbf{k} \tag{6b}$$
$$V_{\mathbf{xx}} = Q_{\mathbf{xx}} - \mathbf{K}^{\mathsf{T}} Q_{\mathbf{uu}} \mathbf{K}. \tag{6c}$$

The backward pass begins by initializing the Value function with the terminal cost and its derivatives $V_N = \ell_f(\mathbf{x}_N)$, and then recursively computing (5) and (6).

C. Line Search

Once the backward pass is completed, the proposed locally-linear policy is evaluated with a forward pass:

$$\hat{\mathbf{x}}_0 = \mathbf{x}_0 \tag{7a}$$

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \alpha\mathbf{k}_i + \mathbf{K}_i(\hat{\mathbf{x}}_i - \mathbf{x}_i) \tag{7b}$$

$$\hat{\mathbf{x}}_{i+1} = \mathbf{f}(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i), \tag{7c}$$

where α is a backtracking search parameter, set to 1 and then iteratively reduced. Finally, this backward-forward process is repeated until convergence to the (locally) optimal trajectory.

D. Complexity and Regularization

The step taken by DDP corresponds to a Newton-Raphson step on the whole unconstrained optimal-control problem [14]. Although DDP searches in the space of control trajectories $\mathbf{U} \in \mathbb{R}^{m\times N}$, it solves the m-dimensional problem N times, not a single problem of size mN. The difference is made stark when considering N Hessians of size $m\times m$ rather than a large $Nm\times Nm$ matrix, as in the direct representation. Since factorization complexity is cubic in the dimension, the respective complexities are $O(Nm^3)$ and $O(N^3m^3)$.

As with all second-order methods, in order to guarantee a descent direction, regularization must be used when the Hessian loses positive definiteness. Typically, a Tikhonov regularization term is added when inverting $Q_{\mathbf{uu}}$ in (5). When the costs are least-square residuals $c(\mathbf{x},\mathbf{u}) = \frac{1}{2}\|\mathbf{r}(\mathbf{x},\mathbf{u})\|^2$, then the Hessians of $\ell_{\mathbf{xx}}$, $\ell_{\mathbf{ux}}$ and $\ell_{\mathbf{uu}}$ are approximated by the square of the Jacobian ($\ell_{\mathbf{xx}} \approx \mathbf{r}_\mathbf{x}^\mathsf{T}\mathbf{r}_\mathbf{x}$ etc.), while the Hessians of $\mathbf{f}$ are neglected. This approximation corresponds to the Gauss-Newton variation and is referred as iLQR or iLQG [16]. The regularization parameter and the descent step length α are adapted online following a Levenberg-Marquardt heuristic. Finally, the suboptimal solution obtained after a fixed number of iterations (typically 1) can be used immediately in a Model Predictive Control setting [23].

III. Control Limits

Due to the strict feasibility property of indirect methods, inequality constraints on the state are handled automatically under the condition that $\mathbf{f}$ maintains regularity, which is obtained by smoothing hard constraints like rigid contacts [24]. This has been shown to be very efficient, even in non-smooth situations like bipedal locomotion, both with direct [25] and indirect [23] optimization. However, the same solution cannot be applied directly to handle inequality constraints on the control. This is an important drawback, as the control might be the joint torques (limited by the motor limits [26]), air pressure or valve aperture of pneumatic robot [27], or as shown in the experiments robot reference angles (limited by the joint range).

In the following, we consider inequality constraints of the form:

$$\underline{\mathbf{b}} \leqslant \mathbf{u} \leqslant \bar{\mathbf{b}} \tag{8}$$

with elementwise inequality and $\underline{\mathbf{b}}, \bar{\mathbf{b}}$ the respective lower and upper bounds. The box constraint accurately describes nearly any set of standard mechanical actuators, and will allow us to use a specialized active-set algorithm which is more efficient and easier to implement. A box-constraint solver can be immediately generalized to any linear inequality constraints using slack variables [28].

In the next sections, two classical ways to enforce the control limits are formalized. These easy-to-implement heuristics will be shown to have significant drawbacks. The last section presents our original solution.

A. Naïve Clamping

A first attempt to enforce box constraints is to clamp the controls in the forward-pass. The element-wise clamping, or projection operator, is denoted by the double square brackets $[\![\cdot]\!]_\mathbf{b}$:

$$[\![\mathbf{u}]\!]_\mathbf{b} = \min(\max(\mathbf{u}, \underline{\mathbf{b}}), \bar{\mathbf{b}}),$$

It is tempting to simply replace (7b) in the forward-pass with

$$\hat{\mathbf{u}}_i = [\![\mathbf{u}_i + \alpha\mathbf{k}_i + \mathbf{K}_i(\hat{\mathbf{x}}_i - \mathbf{x}_i)]\!]_\mathbf{b}, \tag{9}$$

however the corresponding search direction may not be a descent direction anymore, harming convergence. Clamping can also be introduced to the control modification $\mathbf{k}$ in (5):

$$\mathbf{k} \leftarrow [\![\mathbf{k} + \mathbf{u}]\!]_\mathbf{b} - \mathbf{u},$$

and it might also seem sensible that the rows of $\mathbf{K}$ corresponding to clamped controls should be nullified, since the feedback is inactive in these dimensions. Though this might seem reasonable and intuitive, it is demonstrated in the experimental section to be very inefficient.

B. Squashing Functions

Another way to enforce box constraints is to introduce a sigmoidal squashing function $\mathbf{s}(\mathbf{u})$ on the controls

$$\mathbf{x}_{i+1} = \mathbf{f}(\mathbf{x}_i, \mathbf{s}(\mathbf{u}_i)) \tag{10}$$

where $\mathbf{s}()$ is an element-wise sigmoid with the vector limits

$$\lim_{\mathbf{u}\to-\infty} \mathbf{s}(\mathbf{u}) = \underline{\mathbf{b}} \quad \lim_{\mathbf{u}\to\infty} \mathbf{s}(\mathbf{u}) = \bar{\mathbf{b}}.$$

For example $\mathbf{s}(\mathbf{u}) = \frac{\bar{\mathbf{b}}-\underline{\mathbf{b}}}{2}\tanh(\mathbf{u}) + \frac{\bar{\mathbf{b}}+\underline{\mathbf{b}}}{2}$ is such a function. A cost term should be kept on the original $\mathbf{u}$ and not only on the squashed $\mathbf{s}(\mathbf{u})$, otherwise it will reach very high or low values and get stuck on the plateau (see Section IV-B for a practical discussion). An intuition for the poor practical performance of squashing is given by the non-linearity of the sigmoid. Since the backward pass uses a locally quadratic approximation of the dynamics, significant higher order terms will always have a detrimental effect on convergence.

C. Proposed Algorithm

1) Problem Formulation: Clamping does not produce satisfying results since the clamped directions are not taken into account during the inversion of $Q_{\mathbf{uu}}$. On the other hand, squashing introduces an artificial non-linearity in the saturated directions in a way that prevents good performance. We propose to directly take into account the control limits

while minimizing the quadratic model of Q, which amounts to solving a quadratic program (QP) subject to the box constraints (8) at each timestep. The problem is written:

$$\begin{aligned} \underset{\delta\mathbf{u}}{\text{minimize}} \quad & Q(\delta\mathbf{x}, \delta\mathbf{u}) \\ \text{subject to} \quad & \underline{\mathbf{b}} \leqslant \mathbf{u} + \delta\mathbf{u} \leqslant \bar{\mathbf{b}} \end{aligned} \tag{11}$$

The QP is a well understood problem with many methods of solution [29] which together form the backbone of the Sequential-QP approach to nonlinear optimization. When choosing an appropriate solver, two characteristics of the problem at hand should be considered. First, thanks to the Bellman principle, we are solving several small QP's rather than a single big one. Second, since each QP along the backward pass is similar to the next one, an algorithm that can enjoy warm starts should be preferred. The warm-start requirement rules out some classes of algorithms, for example interior-point methods. Since these methods glide smoothly to the solution from the interior, they do not benefit from being initialized at the boundary. Standard active-set methods do traverse the boundary and can be warm-started, but account separately for each constraint activation/deactivation.

2) Proposed Solution: The Projected-Newton class of algorithms are a sub-class of active set methods which were developed for problems with simple constraints, where the projection operator is trivial – like clamping in the case of the box. Their key feature is the projected line-search, whereby the search-point is continuously clamped, allowing multiple constraints to form and break in each iteration. In [30], Bertsekas analyses these methods and proves convergence for a large class of approximate Hessians. In the following we describe a special case thereof, which uses the exact Hessian at all times. Its key feature, which we prove in the Appendix, is that if the initial point has the same active constraint set as the optimum, the solution will be reached in a single iteration. Previous work on incorporating control limits in DDP [17][19], made reference to generic QP algorithms and did not take into account the considerations detailed above.

Since $\delta\mathbf{x}$ is not known during the backward pass, the QP needs to compute both the feedfoward and feedback gains $\mathbf{k}$ and $\mathbf{K}$. The first is obtained directly as the optimum of

$$\begin{aligned} \mathbf{k} = \underset{\delta\mathbf{u}}{\text{argmin}} \quad & \frac{1}{2}\delta\mathbf{u}^{\mathsf{T}} Q_{\mathbf{uu}} \delta\mathbf{u} + Q_{\mathbf{x}}^{\mathsf{T}} \delta\mathbf{u} \\ \text{subject to} \quad & \underline{\mathbf{b}} \leqslant \mathbf{u} + \delta\mathbf{u} \leqslant \bar{\mathbf{b}} \end{aligned}$$

However, we require that out QP solver also return the decomposition of the free dimensions of $Q_{\mathbf{uu}}$, denoted by $Q_{\mathbf{uu},\mathsf{f}}$. This decomposition is used to compute the optimal feedback gain $\mathbf{K}_\mathsf{f} = -Q_{\mathbf{uu},\mathsf{f}} Q_{\mathbf{ux}}$. It follows that $\mathbf{K}_\mathbf{c}$, the rows of $\mathbf{K}$ corresponding to clamped controls, are identically zero. See Appendix I for more details.

3) Complexity: In problems with elaborate dynamics, the effort required to compute the derivatives in the RHS of (4) is often significantly larger than that required for the backward-pass. In that case the extra effort required by the box-QP solver will go unnoticed. If however we ignore the time required for the derivatives, or make it very small by computing them in parallel, the leading complexity term comes from the Cholesky factorization the Projected Newton solver, which is $O(m^3)$. Since standard DDP requires one factorization anyway in (5), the question is how many extra factorizations on average does the box-QP solution impose. The algorithm performs a factorization whenever the active set changes, which might not be often, depending on the problem (see e.g. the middle row of Figure 2). As reported below, in our experiments the average number of factorizations was never larger than 2.

IV. Results

We begin with an initial comparison of the three solution types on a set of simple linear systems randomly selected in Sec. IV-A. We then compare the behavior of squashing and quadratic programming on a nonholonomic car problem in Sec. IV-B. Although for this simple problem analytical optimality can be derived [31], the numerical solution provides an interesting and generic way to control it. Finally, we demonstrate box-DDP on a complex platform, the humanoid robot HRP-2. All the experiments are performed in simulation, which is enough to demonstrate the relationship with respect to unconstrained classical DDP. Applying DDP (and MPC at large) to complex systems such as HRP-2 remains one of the most exciting perspective of this work, which we will discuss in the conclusion.

A. Linear-Quadratic problems

The finite-horizon Linear-Quadratic (LQ) optimal control problem is solved by exactly one full iteration of DDP. When constraints are added, several iteration are necessary. It is described by linear dynamics:

$$\mathbf{x}_{i+1} = \mathbf{f}_\mathbf{x}\mathbf{x}_i + \mathbf{f}_\mathbf{u}\mathbf{u}_i.$$

and the quadratic optimization criterion

$$\underset{\mathbf{U}}{\text{minimize}} \quad \tfrac{1}{2}\mathbf{x}_N^{\mathsf{T}} \ell_{f,\mathbf{xx}} \mathbf{x}_N + \tfrac{1}{2}\sum_{i=0}^{N-1} \left(\mathbf{x}_i^{\mathsf{T}} \ell_{\mathbf{xx}} \mathbf{x}_i + \mathbf{u}_i^{\mathsf{T}} \ell_{\mathbf{uu}} \mathbf{u}_i\right).$$

We generated random LQ problems as follows. The state dimension n was drawn uniformly from $\{10 \dots 100\}$. The control dimension m was drawn from $\{1 \dots \lfloor \frac{n}{2} \rfloor\}$. For a time-step h the random dynamics matrices were $\mathbf{f}_\mathbf{x} = \mathbf{I}_n + h\,\mathbf{N}(n,n)$ and $\mathbf{f}_\mathbf{u} = h\,\mathbf{N}(n,m)$, where $\mathbf{N}$ is a matrix with standard normally distributed elements $\mathbf{N}_{ij} \sim \mathcal{N}(0;1)$, and $\mathbf{I}$ is the identity. The cost matrices were $\ell_{\mathbf{xx}} = \ell_f = h\mathbf{I}_n$ and $\ell_{\mathbf{uu}} = c_u h \mathbf{I}_m$ with c_u the control-cost coefficient. Control bounds were $\underline{\mathbf{b}} = -\mathbf{1}_m$ and $\bar{\mathbf{b}} = \mathbf{1}_m$. The initial state was drawn from the normal distribution $\mathbf{x}_0 = \mathbf{N}(n,1)$.

The bottom row of Figure 2 shows a comparison between the clamping and squashing heuristics and the proposed algorithm. The clamping barely converges to any optimum. The squashing demonstrates a sub-linear convergence. The box-QP solution shows a very characteristic quadratic convergence. Quadratic convergence, which amounts to convergence like Newton method means the doubling of correct significant bits in the solution with each iteration. This manifests as quadratic-looking traces on a log-plot of the

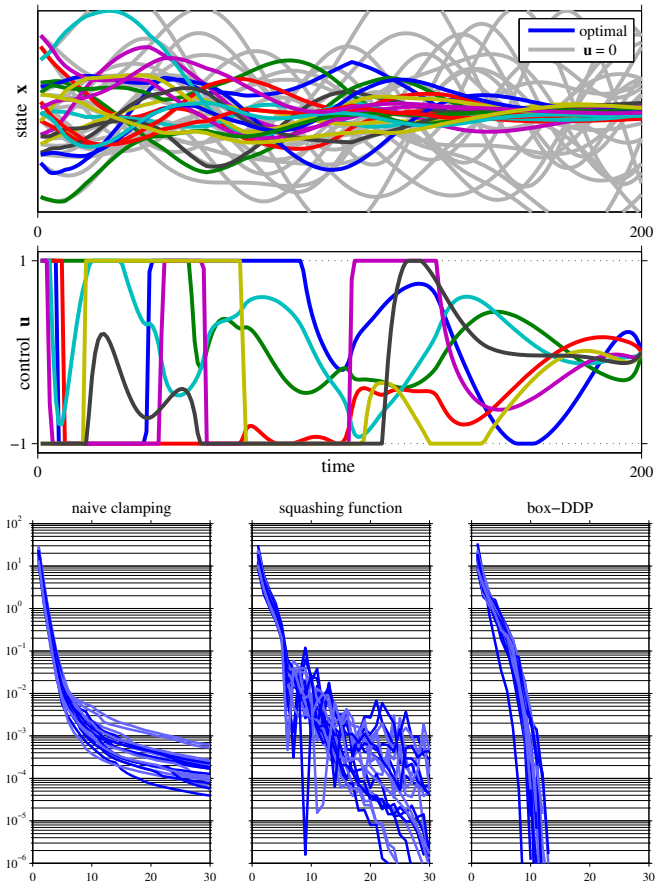

Fig. 2. Control-bounded random linear systems. Here $h = 0.01$, $n = 20$, $m = 7$ and $N = 200$. ***Top:*** A typical state trajectory $\mathbf{X} \equiv \{\mathbf{x}_0 \dots \mathbf{x}_{200}\}$, the passive dynamics ($\mathbf{U} = \mathbf{0}$) are shown in gray. ***Middle:*** the control trajectory $\mathbf{U} \equiv \{\mathbf{u}_0 \dots \mathbf{u}_{199}\}$. The limits $\underline{\mathbf{b}} = -\mathbf{1}$, $\overline{\mathbf{b}} = \mathbf{1}$ are indicated. ***Bottom:*** Cost decrease ΔJ as a function of algorithm iterations during the solution of 20 random LQ problems using the three methods described. ***Left:*** Clamping barely converges to the minimum. ***Center:*** Squashing displays sublinear convergence. ***Right:*** The proposed box-DDP algorithm exhibits quadratic convergence.

convergence trace, as seen in Figure 2. The average number of factorizations per iteration was 1.5. To see why this number is so small, observe the low frequency of constraint-set changes in the middle row of the figure.

B. Car Parking

For the car-like robot, one of the control variables, the angle of the front wheels, is a kinematic, rather than a dynamic variable. When controls specify kinematic variables, bounds arise naturally from the geometry of the problem rather than from actuator limits. This makes kinematic problems an important class for our proposed algorithm.

(x, y, θ, v) is the 4-dimensional state. x, y is the position of the point midway between the back wheels. θ is the angle of the car relative to the x-axis. v is the velocity of the front wheels. The two control signals are ω the front wheel angle and a the front wheel acceleration. For Euler dynamics with a time-step h and letting d denote the distance between the front and back axles, the rolling distance of the front and back wheels are respectively

$$f = hv \tag{13a}$$

$$b = f\cos(\omega) + d - \sqrt{d^2 - f^2\sin^2(\omega)}, \tag{13b}$$

and the h-step dynamics are

$$x' = x + b\cos(\theta) \tag{13c}$$

$$y' = y + b\sin(\theta) \tag{13d}$$

$$\theta' = \theta + \sin^{-1}(\sin(\omega)\frac{f}{d}) \tag{13e}$$

$$v' = v + ha. \tag{13f}$$

The "parking" task is encoded as a final-cost on the distance of the last state from (0,0,0,0), i.e. at the plane origin, facing east and motionless. Distance was measured using the Huber-type function $z(x, p) = \sqrt{x^2 + p^2} - p$. This function is roughly quadratic in a p-sized neighborhood of the origin and linear thereafter. The state cost is

$$\ell_f(\mathbf{x}) = z(x, p_x) + z(y, p_y) + z(\theta, p_\theta) + z(v, p_v)$$

We chose $p_x = p_y = 0.1m$, $p_\theta = 0.01rad$ and $p_v = 1m/s$ to compensate for the relative difficulty of changing each variable. Because it is easier to stop the car ($v = 0$) than to orient it ($\theta = 0$), we would like the optimizer to focus on the harder task once near enough to the goal-state. A running cost is added to penalize cartesian distance from the origin

$$\ell(\mathbf{x}) = 0.01\big(z(x, p_x) + z(y, p_y)\big)$$

This term encourages parking maneuvers which do not take the car far from the origin. $\ell(\mathbf{u}) = c_\omega\omega^2 + c_a a^2$ with $c_\omega = 0.01$ and $c_a = 0.0001$. Cost coefficients were chosen to be small in order to encourage the controller to hit the bounds $b_\omega = \pm 0.5rad$ and $b_a = \pm 2m/s^2$.

Since the "clamping" heuristic performed so badly in the previous case, here we used the car parking domain to compare box-DDP only to the "squashing" heuristic. The squashing function used was

$$\omega(\tilde{\omega}) = 0.5 \times \tanh(\tilde{\omega})$$
$$a(\tilde{a}) = 2 \times \tanh(\tilde{a}).$$

In order to prevent the "pre-controls" $(\tilde{\omega}, \tilde{a})$ from diverging, a small explicit cost on these was added

$$\ell(\tilde{\omega}, \tilde{a}) = c_\omega\omega^2 + c_a a^2 + 10^{-6}(\tilde{\omega}^2 + \tilde{a}^2).$$

This additional term is small enough to not significantly modify the problem, but large enough to pull $(\tilde{\omega}, \tilde{a})$ back towards the origin when they are too large.

Fig. 3 compares the results obtained with the two solvers. Similar trajectories are obtained, but with much higher gains $\mathbf{k}, \mathbf{K}$ in the squashing case. Fig. 4 gives the convergence rate comparison. The squashing-function solution barely converge while the box DDP converges quadratically.

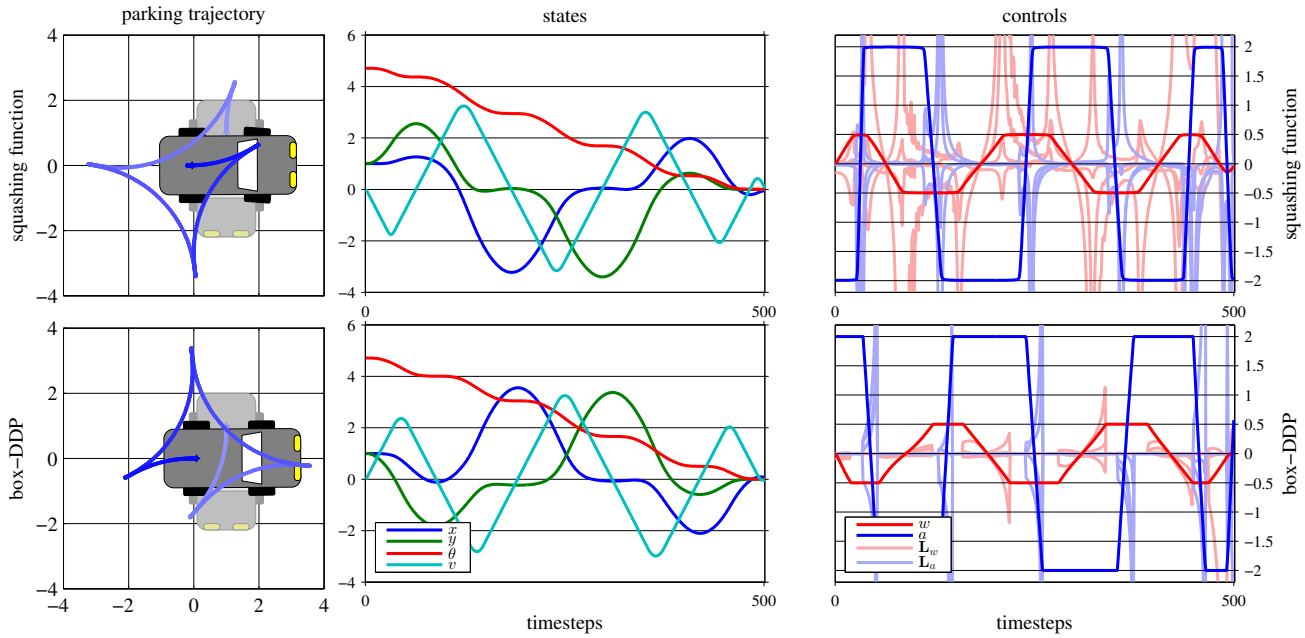

Fig. 3. Comparison between squashing-functions and BOX-DDP. ***Left column:*** Bird's eye view of the parking trajectories obtained after convergence of both DDP, starting from $(1, 1, \frac{3\pi}{2}, 0)$ (gray car, background) and ending at $(0, 0, 0, 0)$ (foreground). Both trajectories are approximations of two different global optima (the first being the reflection of the second through the first bisector). ***Middle column:*** Corresponding state trajectories $(x, y, \theta, v)_i$ plotted over time. ***Right column:*** Controls (ω, a) (solid) and feedback gains $\mathbf{K}$ (light colors). The feedback gains of the squashing heuristic are much stronger to overcome the sigmoid slope, and explain the worse converge behavior. Note how for the BOX-DDP solution, the rows of the 2×4 matrices $\mathbf{K}$ vanish whenever the corresponding controls are clamped, as expected from the discussion in Sec III-C.2.

C. Humanoid robot

Like many modern full-size humanoid robots, HRP-2 [32] is powered by direct-current electrical motors coupled with high-ratio gears (typically, harmonic drive with ratio $1/200$) which make it very stiff. Two solutions are possible to apply the DDP on a robot such as HRP-2. The first one is to perform precise system identification, taking into account the well-known motor dynamics, the PD-controller transfer function and the harmonic gear frictions. The inverse of this model provides a feed-forward torque control input [33]. However, despite some recent work in this direction [34], direct feed-forward current control is not yet a functional option, while the lack of joint torque sensor on most of humanoid robots prevent feedback torque control. Alternatively, the low-level PD controllers of the robot can be modeled inside the forward dynamics. The control input $\mathbf{u}$ is then the reference joint angle. This solution is appealing, since the PD controllers can be considered strong enough to nullify the gear dry friction, which need not be modelled. The control is then limited by the joint range, which should not be hit as it would likely damage the robot. For this reason box-DDP is appealing since the joint references output by the algorithm are guaranteed to be inside the limits.

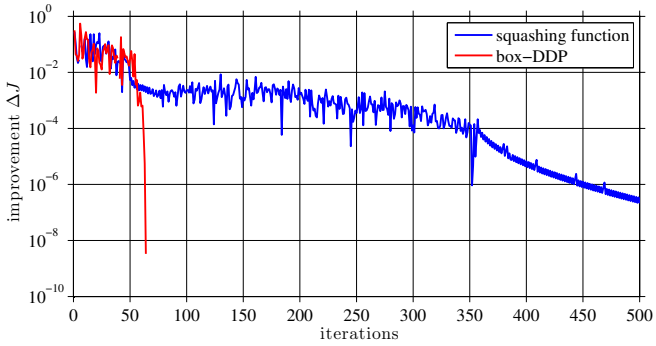

Fig. 4. Cost decrease ΔJ of the two algorithms for the car-parking problem. Box-DDP converges quadratically after 64 iterations, while the squashing-function solution has barely converged by 500.

Optimal control allows for very simple specification of the robot movement. In the demonstrated example, the robot has to reach a moving target with its right gripper, while standing and if necessary stepping to maintain its balance. Several cost functions are used to define various aspects of the motion of the robot. The balance is enforced by setting three cost functions: on the chest and pelvis angles θ to keep them horizontal; on the chest altitude z; and on the capture point $\mathbf{a}$ to keep it on the line between the feet. The two last ones penalize the linear and angular momenta:

$$\begin{aligned}\ell_{bal}(\mathbf{x}) =& c_\theta \left(\|\theta_{pelvis}(\mathbf{x})\|^2 + \|\theta_{chest}(\mathbf{x})\|^2 \right) \\ &+ c_{\mathbf{z}}(z_{chest}(\mathbf{x}) - z^*)^2 + c_\alpha z(\mathbf{a}(\mathbf{x}) - \mathbf{a}^\perp(\mathbf{x}))\end{aligned}$$

with z^* the initial chest altitude, $\mathbf{a}^\perp$ the orthogonal projection of $\mathbf{a}$ on the line between the ankles, $c_\theta = 0.3$, $c_z = 0.2$ and $c_a = 1$. The stepping is emphasized by putting a cost to keep the feet parallel to the ground and oriented toward the target:

$$\ell_{step}(\mathbf{x}) = c_{roll} \|\theta_{lf,rf}(\mathbf{x})\|^2 + c_{yaw}(\gamma_{feet}(\mathbf{x}) - \gamma^*)^2$$

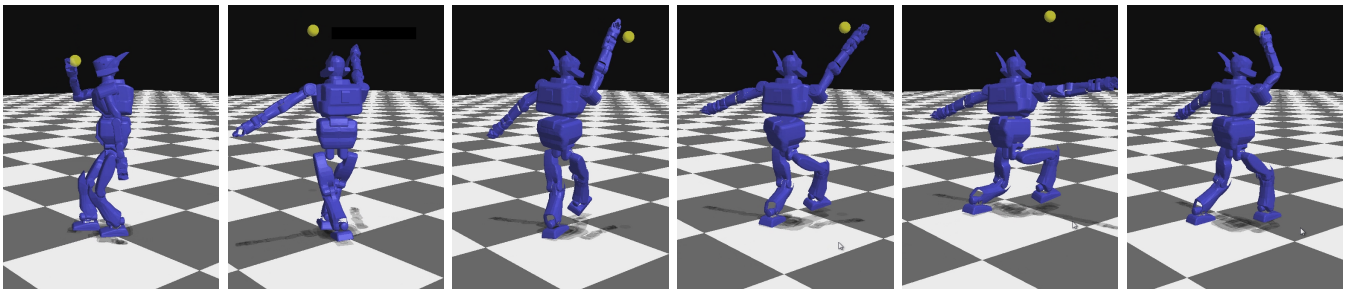

Fig. 5. Reaching a moving target, stepping when necessary. A sequence of frames of full body motion synthesized in real-time for the HRP2 robot. The robot is trying to reach the moving target while stabilizing itself. When the rotation causes the legs to collide, the robot stumbles, takes a step to balance itself, and reaches for the target once more.

with $\theta_{lf,rf}$ the angle of the feet with respect to the ground, γ_{feet} the yaw angle of the line between the two ankles, γ^* the yaw angle of the target in the egocentric cylindrical coordinates, $c_{roll} = 0.05$ and $c_{yaw} = 0.1$. Finally, the reaching is triggered by a cost on the distance to the target:

$$\ell_{reach} = c_{reach}\|\mathbf{p}(\mathbf{x}) - \mathbf{p}^*\|^2$$

with $\mathbf{p}$ the gripper position and $\mathbf{p}^*$ the target position. All the cost are squares of residuals, which enable us to use the Gauss-Newton approximation of the second derivatives. This is an important shortcut since the second order derivatives would be very expensive to compute with a system of the size of HRP-2. Collision avoidance is enforced by the simulator in the forward pass.

An overview of the obtained motion is given in Fig. 5. The complete motion, along with a set of other examples, are displayed on the companion video.

V. Conclusion

This paper proposed a modification of the DDP which allows us to incorporate control limits. This is a key feature for applying the DDP algorithm to real robots. In particular, it is mandatory when the control input specifies some kinematic variables, like the steering direction of the car or the joint references of the humanoid. Our solution is very fast and keeps the good convergence properties of the DDP algorithm. It is also exact, in the sense that the specified constraint can not be exceeded in any situation. It enables us to control the humanoid robot HRP-2 in real time with a desktop personal computer in simulation, while interacting with it using an haptic device. The next step is to apply the same control scheme on the real HRP-2 robot. The key point for that is to introduce some feedback terms in addition to the state estimation to make the MPC behavior more robust to modeling errors.

Acknowledgment

We thank Akshay Srinivasan for his useful insights. This work was supported by the National Science Foundation.

Appendix I
Projected-Newton QP Solution

Consider the generic problem:

$$\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{H}\mathbf{x} + \mathbf{q}^\mathsf{T}\mathbf{x} \tag{14a}$$

$$\text{subject to} \quad \underline{\mathbf{b}} \leqslant \mathbf{x} \leqslant \bar{\mathbf{b}} \tag{14b}$$

The algorithm proceeds by iteratively identifying the active constraints, and then performing a projected Newton step using the reduced Hessian in the free sub-space. Begin at some feasible initial guess $\mathbf{x} = [\![\mathbf{x}]\!]_\mathbf{b}$ and define the gradient $\mathbf{g} = \nabla_\mathbf{x} f = \mathbf{q} + \mathbf{H}\mathbf{x}$. The complimentary sets of *clamped* and *free* indices c and f are

$$\mathsf{c}(\mathbf{x}) = \left\{ j \in 1\ldots n \;\middle|\; \begin{array}{ll} \mathbf{x}_j = \underline{\mathbf{b}}_j\,, & \mathbf{g}_j > 0 \\ \text{or} & \\ \mathbf{x}_j = \bar{\mathbf{b}}_j\,, & \mathbf{g}_j < 0 \end{array} \right\} \tag{15a}$$

$$\mathsf{f}(\mathbf{x}) = \left\{ j \in 1\ldots n \;\middle|\; j \notin \mathsf{c} \right\} \tag{15b}$$

For readability, we sort the index partition $\{\mathsf{f}, \mathsf{c}\}$:

$$\mathbf{x} \leftarrow \begin{bmatrix} \mathbf{x}_\mathsf{f} \\ \mathbf{x}_\mathsf{c} \end{bmatrix}, \quad \mathbf{q} \leftarrow \begin{bmatrix} \mathbf{q}_\mathsf{f} \\ \mathbf{q}_\mathsf{c} \end{bmatrix}, \quad \mathbf{H} \leftarrow \begin{bmatrix} \mathbf{H}_{\mathsf{ff}} & \mathbf{H}_{\mathsf{fc}} \\ \mathbf{H}_{\mathsf{cf}} & \mathbf{H}_{\mathsf{cc}} \end{bmatrix}, \tag{16}$$

The gradient in the free subspace is

$$\mathbf{g}_\mathsf{f} = \nabla_{\mathbf{x}_\mathsf{f}} f = \mathbf{q}_\mathsf{f} + \mathbf{H}_{\mathsf{ff}}\mathbf{x}_\mathsf{f} + \mathbf{H}_{\mathsf{fc}}\mathbf{x}_\mathsf{c},$$

The Newton step in the free subspace is then:

$$\Delta\mathbf{x}_\mathsf{f} = -\mathbf{H}_{\mathsf{ff}}^{-1}\mathbf{g}_\mathsf{f} = -\mathbf{H}_{\mathsf{ff}}^{-1}\left(\mathbf{q}_\mathsf{f} + \mathbf{H}_{\mathsf{fc}}\mathbf{x}_\mathsf{c}\right) - \mathbf{x}_\mathsf{f}.$$

The full step is therefore:

$$\Delta\mathbf{x} = \begin{bmatrix} \Delta\mathbf{x}_\mathsf{f} \\ \mathbf{0}_\mathsf{c} \end{bmatrix}. \tag{17}$$

The projected Newton candidate point $\hat{\mathbf{x}}$ for a line-search parameter α is

$$\hat{\mathbf{x}}(\alpha) = [\![\mathbf{x} + \alpha\Delta\mathbf{x}]\!]_\mathbf{b}. \tag{18}$$

A backtracking line-search reduces α until the Armijo condition [35] is satisfied

$$\frac{f(\mathbf{x}) - f(\hat{\mathbf{x}}(\alpha))}{\mathbf{g}^\mathsf{T}\left(\mathbf{x} - \hat{\mathbf{x}}(\alpha)\right)} > \gamma \tag{19}$$

with $0 < \gamma < \frac{1}{2}$ the minimally acceptable reduction ratio. We use the oft-quoted $\gamma = 0.1$ in the experiments. By design, the

Algorithm I $\mathbf{x}^* \longleftarrow \mathrm{QP}[\mathbf{H}, \mathbf{q}, \underline{\mathbf{b}}, \bar{\mathbf{b}}, \mathbf{x}]$

Repeat until convergence:

1) Get indices: Equations (15).

2) Get Newton step: Equations (16).

3) Convergence: If $\|\mathbf{g}_\mathsf{f}\| < \epsilon \ll 1$, terminate.

4) Line search: Decrease α in (18) until (19) is satisfied. Accept the candidate $\mathbf{x} \leftarrow \hat{\mathbf{x}}(\alpha)$.

key feature of the algorithm is the following:

Lemma. *If the initial point* $\mathbf{x}$ *has the same clamped constraints as the optimum* $\mathsf{c}(\mathbf{x}) = \mathsf{c}(\mathbf{x}^*)$*, then the solution will be reached in a single iteration.*

Proof. Setting $\Delta\mathbf{x}_\mathsf{f} = 0$ at the optimum, we have from (17) that $\mathbf{x}_\mathsf{f}^* = -\mathbf{H}_{\mathsf{ff}}^{-1}(\mathbf{q}_\mathsf{f} + \mathbf{H}_{\mathsf{fc}}\mathbf{x}_\mathsf{c})$. If $\mathsf{c}(\mathbf{x}) = \mathsf{c}(\mathbf{x}^*)$ then $\mathbf{x}_\mathsf{c} = \mathbf{x}_\mathsf{c}^*$ and therefore $\Delta\mathbf{x}_\mathsf{f} = \mathbf{x}_\mathsf{f}^* - \mathbf{x}_\mathsf{f}$ taking us directly to the minimum in one step. □

REFERENCES

[1] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[2] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *The International Journal of Robotics Research*, vol. 3, no. 1, pp. 43–53, 1987.

[3] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[4] N. Mansard, O. Khatib, and A. Kheddar, "Integrating unilateral constraints inside the stack of tasks," *IEEE Trans. on Robotics*, vol. 25, no. 11, pp. 2493–2505, 2009.

[5] P. Baerlocher, "Inverse kinematics techniques for the interactive posture control of articulated figures," Ph.D. dissertation, EPFL, 2001.

[6] L. Sentis, "Synthesis and control of whole-body behaviors in humanoid systems," Ph.D. dissertation, Stanford University, 2007.

[7] N. Mansard, O. Stasse, F. Chaumette, and K. Yokoi, "Visually-guided grasping while walking on a humanoid robot," in *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, 2007, pp. 3041–3047.

[8] S. Hak, N. Mansard, O. Stasse, and J.-P. Laumond, "Reverse control for humanoid robot task recognition," *IEEE Trans. Sys. Man Cybernetics*, vol. 42, no. 6, pp. 1524–1537, 2012.

[9] S. M. Khansari-Zadeh and A. Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[10] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The mathematical theory of optimal processes*. Interscience New York, 1962.

[11] O. Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, no. 1, pp. 357–373, Dec. 1992.

[12] D. Q. Mayne, "A second-order gradient method of optimizing non-linear discrete time systems," *Int J Control*, vol. 3, p. 8595, 1966.

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

[14] L. Z. Liao and C. A. Shoemaker, "Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems," *Cornell University, Ithaca, NY*, 1992.

[15] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2004, pp. 222–229.

[16] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the American Control Conference (ACC'05)*, Portland, OR, USA, 2005, pp. 300–306.

[17] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach," *Journal of Optimization Theory and Applications*, vol. 2, no. 6, pp. 411–440, Nov. 1968.

[18] D. J. W. Ruxton, *Differential Dynamic Programming and Optimal Control of Quality Constrained Continuous Dynamic Systems*. University of Central Queensland, Department of Mathematics and Computing, 1991.

[19] D. M. Murray and S. J. Yakowitz, "Constrained differential dynamic programming and its application to multireservoir control," *Water Resources Research*, vol. 15, no. 5, p. 10171027, 1979.

[20] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 03, p. 321, Jun. 2008.

[21] M. Diehl, H. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear mpc and moving horizon estimation," *Nonlinear Model Predictive Control*, p. 391, 2009.

[22] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 670–685, 2009.

[23] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'12)*, 2012, pp. 4906–4913.

[24] Y. Tassa and E. Todorov, "Stochastic complementarity for local control of discontinuous dynamics," in *Proceedings of Robotics: Science and Systems (RSS)*, 2010.

[25] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 31, no. 4, p. 43, 2012.

[26] L. Saab, O. Ramos, N. Mansard, P. Souères, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transaction on Robotics*, no. 2, pp. 346–362, April 2013.

[27] Y. Tassa, T. Wu, J. Movellan, and E. Todorov, "Modeling and identification of pneumatic actuators," in *IEEE International Conference Mechatronics and Automation*, August 2013.

[28] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[29] J. Nocedal and S. J. Wright, *Numerical optimization*. New York: Springer, 2006.

[30] D. P. Bertsekas, "Projected newton methods for optimization problems with simple constraints," *SIAM Journal on Control and Optimization*, vol. 20, no. 2, pp. 221–246, Mar. 1982.

[31] P. Souères and J.-D. Boissonnat, "Optimal trajectories for nonholonomic robots," in *Robot Motion Planning and Control*, ser. Lecture Notes in Control and Information Sciences, J.-P. Laumond, Ed. Springer, 1998, vol. 229.

[32] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi, "Design of prototype humanoid robotics platform for hrp," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*, 2002.

[33] T. Yoshikawa and O. Khatib, "Compliant motion control for a humanoid robot in contact with the environment and humans," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, 2008.

[34] S. Traversaro, A. Del Prete, R. Muradore, L. Natale, and F. Nori, "Inertial parameter identification including friction and motor dynamics," in *IEEE-RAS International Conference on Humanoid Robots (Humanoid'13)*, Atlanta, USA, October 2013, [to appear].

[35] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives." *Pacific Journal of Mathematics*, vol. 16, no. 1, pp. 1–3, 1966.