Parametric Trajectory Libraries for Online Motion Planning with Application to Soft Robots

Tobia Marcucci, Manolo Garabini, Gian Maria Gasparri, Alessio Artoni, Marco Gabiccini, and Antonio Bicchi

Abstract In this paper we propose a method for online motion planning of constrained nonlinear systems. The method consists of three steps: the offline generation of a library of parametric trajectories via direct trajectory optimization, the online search in the library for the best candidate solution to the optimal control problem we aim to solve, and the online refinement of this trajectory. The last phase of this process takes advantage of a sensitivity-like analysis and guarantees to comply with the first-order approximation of the constraints even in case of active set changes. Efficiency of the trajectory generation process is discussed and a valid strategy to minimize online computations is proposed; together with this, an effective procedure for searching the candidate trajectory is also presented. As a case study, we examine optimal control of a planar soft manipulator performing a pick-and-place task: through simulations and experiments, we show how crucial online computation times are to achieve considerable energy savings in the presence of variability of the task to perform.

Key words: Direct Trajectory Optimization, Trajectory Libraries, Soft Robots

1 Introduction

Trajectory planning is a problem of critical importance in robotics. Among the many, two fundamental motivations are: increase robot autonomy towards auto-

Tobia Marcucci, Manolo Garabini, Gian Maria Gasparri, Marco Gabiccini, Antonio Bicchi
Research Center "E. Piaggio", Università di Pisa, Pisa, Italy

Tobia Marcucci, Marco Gabiccini, Antonio Bicchi
Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genoa, Italy

Alessio Artoni, Marco Gabiccini
Dipartimento di Ingegneria Civile e Industriale, Università di Pisa, Pisa, Italy

mated discovery of complex behaviors [13], and maximize performances such as energy efficiency or execution speed (critical in production cycles [21] and in mobile robotics [16]).

Trajectory planning is even more important for the novel generation of soft robots. These systems, thanks to the introduction of continuous or lumped elasticity in their design (see [11] and [19] for two recent reviews), present richer dynamics with respect to their rigid counterparts. A suitable exploitation of such characteristic leads to substantial improvements in efficiency [20], and maximum speed [9]. However, in order to achieve these results, two ingredients are fundamental for the control scheme: optimality and feedforward. Optimal control theory is one of the very few tools able to exploit elastic potential energy of soft robots [9, 10], whereas closed-loop controllers may alter the natural dynamics of the system [4]. Additionally, in a great variety of practical situations, problem parameters may vary and be known only at the very beginning of the task, posing the challenging requirement of generating the motion planning in a negligible time with respect to the smallest time constant of the system dynamics.

In this context, due to the dynamic nonlinearities, nonlinear programming is one of the only viable options to both exploit the system dynamics and reduce the need for feedback. However, its direct application is not sufficiently fast, and the need for alternative methods for real-time computation of (near-) optimal trajectories arises.

A classical answer to this problem is Dynamic Programming (DP) where all the computational effort is shifted offline, but the curse of dimensionality makes this method inapplicable to large scale systems. Similar in concept to DP, but different in practice, is the trajectory library approach [15]; here the offline phase consists in the computation of a set of optimal motions and the synthesis of optimal controllers able to steer the system to one of the stored paths. Such libraries can be built with various techniques, e.g. randomized motion planning [17]; in this paper we opt for the mature field of Direct Trajectory Optimization (DTO) [2]. A combination of DTO and differential DP for the trajectory generation was proposed by [12], where neighboring optimal controllers were also stored in the library. In [18] a feedback motion planning algorithm, which uses sum-of-squares verification for the computation of stability regions along the trajectories in the library, was presented.

A different line to approach real-time optimization comes from the field of Nonlinear Model Predictive Control (NMPC) where, in some schemes, only the solution of a computationally lighter approximation of the nonlinear Optimal Control Problem (OCP) is performed. For example, [5] perform only one Sequential Quadratic Program (SQP) iteration per sampling time, whereas [23] use sensitivity analysis to anticipate the online solution of the OCP and subsequently correct the result. These are, however, control techniques and their applicability is limited to local corrections of a nominal trajectory computed at the previous sampling time. From our side, we are interested in optimal feedforward actions without restrictions on the values of the OCP parameters.

In this paper we aim to merge the previous points of view, using NMPC correction techniques for the online refinement of trajectories stored in a library. Furthermore, unlike the works listed above, our goal is not the system stabilization,

which would implicitly identify the system initial state as the OCP parameter, but rather to consider a generic definition of a parametric NonLinear Program (NLP). In this framework, the optimal trajectory might be a function of any parameter in the underlying NLP, such as the system final state or the cost function weights.

The presented method is composed of three phases: the offline generation of a library of parametric trajectories via DTO, the online search in the library for the best candidate solution to the OCP we aim to solve, and the online refinement of this trajectory. The last step consists in a real-time correction based on a sensitivity analysis or in the solution of a Quadratic Program (QP), depending on the magnitude of the parameter variation. Also the selection of the candidate trajectory is based on sensitivity information, which, together with an efficient statement of the QP, are stored in the library.

The main contribution of this work is the development of an algorithm for real-time optimal motion planning; nonetheless, to the best of the authors' knowledge, the experimental results presented in this work are the first successful application of open-loop optimal control of flexible-joint robots, strategy that recently proved to be the most adequate to the control of this type of systems [4].

In Sec. 2 we expose a motivational pick-and-place problem with parametric object mass, initial and terminal pose. We show how soft robots could in principle save a considerable amount of energy (approximately 30%) but, assuming the parameter values to be known only at the very beginning of the task, this saving can be achieved only with real-time motion planning. After a brief review of parametric nonlinear optimization (Sec. 3), in Sec. 4 we present an efficient strategy for the online refinement of optimal trajectories. Sec. 5 describes the method of parametric trajectory libraries, and shows how it overcomes the difficulties outlined in Sec. 2. Experimental validation is presented in Sec. 6 (and in the accompanying video), pointing out the further benefit of this method of not requiring settling times caused by model-plant mismatches. Finally, conclusions are presented in Sec. 7.

2 Motivational Example and Problem Statement

Pick and place is probably the most striking example of soft robots efficiency; however, even for this simple task, if we allow object positions and masses to vary, bringing the robot to resonance becomes a non-trivial operation which requires advanced planning techniques. In this section we analyze energy efficiency of a pick-and-place problem for the planar manipulator shown in Fig. 1 and described in Tab. 1. We show how, in principle, soft actuation could outperform rigid one but excessive planning times make these savings out of reach for common DTO techniques and the need of novel methods for online motion planning arises.

The task is to move $n_\mathrm{o} = 100$ objects with random masses $m_\mathrm{o} \in [0.3, 0.5]$ kg between random positions which are assumed to be known only at the completion of the previous task. We compare rigid actuation (R), soft actuation with variable stiffness (VS), and soft actuation with constant stiffness (CS); including returns, we

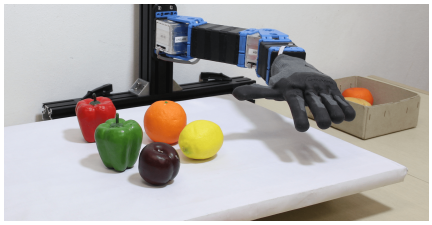


Fig. 1: Planar RR manipulator powered by series elastic actuators with the fast version of the Pisa-IIT SoftHand with a closure time of 0.1 s.

	Mass (kg)	Friction ($\frac{\text{Nms}}{\text{rad}}$)	Center of mass position (m)	Length (m)
Link 1	0.75	0.040	0.20	0.22
Link 2	0.88	0.030	0.19	0.22

	Inertia (mgm^2)	Friction ($\frac{\mu\text{Nms}}{\text{rad}}$)	Reduction ratio
Motor	0.50	0.22	205

Table 1: Parameters of the planar manipulator (identified from a point-mass model).

obtain $2n_\text{o}$ OCPs per actuation type. Preliminarily, we will not take computation times into account assuming that optimal trajectories are computed instantaneously.

Denoting with $\varphi \in \mathbb{R}^{n_\varphi}$ the angular position of the links and with $\theta \in \mathbb{R}^{n_\varphi}$ the link-side angular position of the motors, the soft robot dynamics is modeled as

$$M(\varphi)\ddot{\varphi} + c(\varphi,\dot{\varphi}) + K(\varphi-\theta) = 0, \quad I\ddot{\theta} + F\dot{\theta} + K(\theta-\varphi) = \tau, \tag{1}$$

where $M \in \mathbb{R}^{n_\varphi \times n_\varphi}$ is the link inertia matrix, c includes Coriolis, centrifugal, frictional, and gravitational terms, K is the diagonal stiffness matrix, I is the diagonal inertia matrix of the motors, F is the diagonal friction matrix of the motors, and τ represents motor torques. The dynamics of the corresponding rigid robot is simply

$$(M(\varphi)+I)\ddot{\varphi} + c(\varphi,\dot{\varphi}) + F\dot{\varphi} = \tau. \tag{2}$$

For $k = 1,\dots,2n_\text{o}$, each OCP has the following structure (case-specific equations or variables are explicitly indicated):

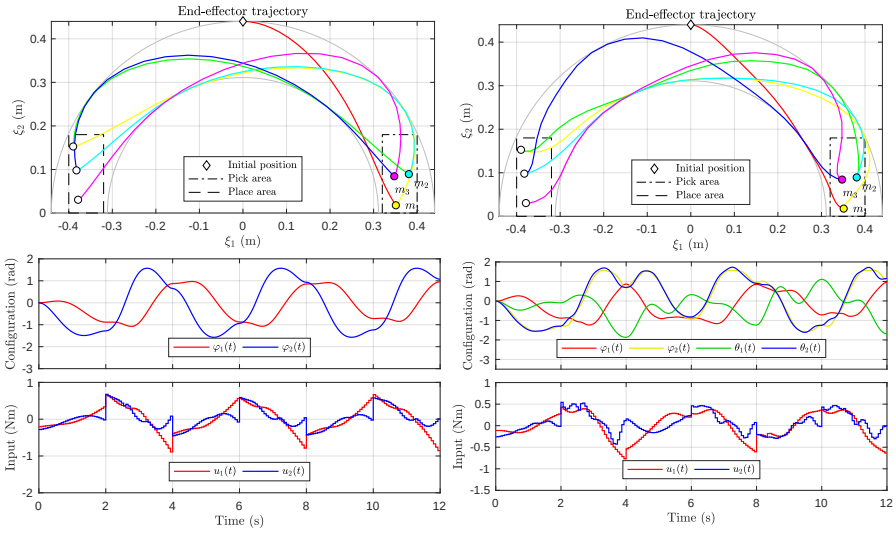
$$\min_{\tau(t),\ \underbrace{K}_{\text{VS}}} \quad J^k := \frac{1}{t_\text{f}^k - t_\text{i}^k}\int_{t_\text{i}^k}^{t_\text{f}^k} \tau^2(t)\,\text{d}t \quad \text{s.t.} \tag{OCP}$$

$$\{\ \underbrace{(1)}_{\text{CS, VS}},\ \underbrace{(2)}_{\text{R}}, \tag{dynamic model}$$

$$\varphi(t_\text{i}^k) = \varphi_\text{i}^k,\ \dot{\varphi}(t_\text{i}^k) = 0,\ \underbrace{\theta(t_\text{i}^k) = \theta_\text{i}^k,\ \dot{\theta}(t_\text{i}^k) = 0}_{\text{CS, VS}}, \tag{initial conditions}$$

$$\xi(\varphi(t_\text{f}^k)) = \xi_\text{f}^k,\ \dot{\varphi}(t_\text{f}^k) = 0,\ \underbrace{\dot{\theta}(t_\text{f}^k) = 0}_{\text{CS, VS}}, \tag{terminal constraints}$$

$$\underline{\varphi} \le \varphi(t) \le \overline{\varphi},\ \underline{\tau} \le \tau(t) \le \overline{\tau},\ \underbrace{\underline{v} \le \dot{\varphi}(t) \le \overline{v}}_{\text{R}},\ \underbrace{\underline{v} \le \dot{\theta}(t) \le \overline{v}}_{\text{CS, VS}},\ \underbrace{K \succ 0}_{\text{VS}}\}, \tag{variable bounds}$$



(a) Rigid robot.

(b) Robot with constant stiffness.

Fig. 2: First three pick-and-place tasks. From top to bottom: end-effector trajectory, configuration angles as functions of time, input torques as functions of time.

where $\xi(\varphi) \in \mathbb{R}^{n_\xi}$ denotes the position of the end-effector, the subscripts i and f represent initial and final values, respectively, whereas underlines and overlines denote lower and upper bounds, respectively. The time horizon of each plan is $t_{\mathrm{f}}^k - t_{\mathrm{i}}^k = 2$ s which, overall, demonstrated to be the shortest horizon allowed by actuator velocity limits for this positioning of the objects. The (OCP) is discretized with a multiple shooting approach: 30 shooting intervals per plan with piecewise constant inputs and 5 integration steps per shooting. The integration scheme is the 4th order explicit Runge-Kutta. Bounds on the optimization variables are: $\overline{\tau} = -\underline{\tau} = 5$ Nm, $\overline{\varphi} = -\underline{\varphi} = \begin{bmatrix}\infty & \pi/2\end{bmatrix}^\top$ rad, $\overline{v} = -\underline{v} = 4$ rad/s. NLPs are solved with `IPOPT` [22] and implemented in the highly efficient framework `CasADi` [1]. Energy efficiency is compared analyzing the mean cost, defined as $\bar{J}^* := \frac{1}{2n_{\mathrm{o}}} \sum_{k=1}^{2n_{\mathrm{o}}} J^{k*}$, with J^{k*} denoting the solution of the kth (OCP).

The mean cost set by the rigid robot (R) is $\bar{J}^*_{\mathrm{R}} = 0.184$ (Nm)^2; by way of illustration, Fig. 2a shows the optimal trajectories for the first three movements. Optimizing the stiffness of each actuator at every movement (VS), we reduce the energy consumption by 31.0% setting $\bar{J}^*_{\mathrm{VS}} = 0.127$ (Nm)^2. However, this analysis neglects energy consumption due to the change (and the retention) of the actuator stiffness that, depending on the motor design, might be significant. To make the comparison fair, we fix motor stiffnesses (CS) to the mean optimal values obtained from the previous analysis $\bar{K}^* = \mathrm{diag}(0.316, 1.772)$ Nm/rad. The resulting average cost is $\bar{J}^*_{\mathrm{CS}} = 0.137$ (Nm)^2, with a satisfactory saving with respect to the rigid case (25.8%)

and a limited loss with respect to the variable stiffness case (-7.6%). Fig. 2b shows the optimal trajectories for the latter case.

Taking computation times into account, however, these conclusions change drastically. Since the final state of the robot is not an equilibrium ($\varphi(t_{\mathrm{f}}^k) \neq \theta(t_{\mathrm{f}}^k)$), delays in the computations and, consequently, in the application of the open-loop control might have catastrophic consequences. NLP solutions, in fact, required on average 0.853 s for the constant stiffness case and 1.658 s for variable stiffness case, making these optimal trajectories useless in practice. (Computations are performed with a notebook computer with the 2.4 GHz Intel Core i7 processor and a 16 GB 1867 MHz LPDDR3 memory.)

The problem address by this work is to extend the scope of DTO methods to the real-time motion planning of dynamical systems. The goal is not to define a stabilizing policy for the system under analysis, but rather to derive a general-purpose strategy able to react online to environment changes and variations in the task parameters. To this end, we decide to leverage on offline computations and we consider a limited loss in optimality an admissible compromise but, on the other hand, we are as intransigent as possible towards the feasibility of the synthesized motions. The algorithm we propose in this paper is able to generate nearly optimal (and first-order feasible) trajectories in an amount of time that, for the case study presented in this section, is two orders of magnitude lower than the one obtained with the state-of-the-art NLP techniques mentioned above.

3 Notions of Nonlinear Optimization

In this section we review some basic concepts of parametric nonlinear optimization. Even if these are well known (see, e.g., [14] and [7]), they are reviewed here for convenience as they are the basis for the developments in the next sections. In the following, we will denote with $\mathrm{d}_x f(x)$ ($\partial_x f(x)$) the total (partial) derivative of $f(x)$ with respect to x.

Direct trajectory optimization techniques translate the continuous time (OCP) into a finite dimensional NLP. Including the explicit dependence on a set of parameters p, we get a multiparametric NLP, which can be stated as

$$\min_x f(x,p) \text{ s.t. } \{g(x,p) = 0,\ h(x,p) \geq 0\}, \tag{mpNLP}$$

where $x \in \mathbb{R}^{n_x}$, $p \in \mathbb{R}^{n_p}$, $f \in \mathbb{R}$, $g \in \mathbb{R}^{n_g}$, and $h \in \mathbb{R}^{n_h}$. For a feasible x, we call $\mathscr{A}(x,p) := \{k \in \{1,\dots,n_h\} | h_k(x,p) = 0\}$ active set, and we distinguish active inequalities $h_{\mathrm{a}}(x,p) := \{h_k(x,p) | k \in \mathscr{A}(x,p)\} \in \mathbb{R}^{n_{\mathrm{a}}}$ from inactive inequalities $h_{\mathrm{i}}(x,p) := \{h_k(x,p) | k \in \{1,\dots,n_h\} \setminus \mathscr{A}(x,p)\} \in \mathbb{R}^{n_h - n_{\mathrm{a}}}$. Moreover, denoting with $x^*(p)$ a local solution of (mpNLP), we define the optimal active set $\mathscr{A}^*(p) := \mathscr{A}(x^*(p),p)$.

The Lagrangian function for (mpNLP) is $l(x,\lambda,\mu,p) := f(x,p) - \lambda^\top g(x,p) - \mu^\top h(x,p)$, with Lagrange multipliers $\lambda \in \mathbb{R}^{n_g}$ and $\mu \in \mathbb{R}^{n_h}$. The following theorem states necessary conditions for x to be a local solution of (mpNLP).

Theorem 1 (First-Order Necessary Conditions (FONC)). *Consider the set of Karush-Kuhn-Tucker (KKT) equations*

$$\partial_x^\top l(x,\lambda,\mu,p) = 0, \qquad \textit{(stationarity)} \tag{3a}$$
$$g(x,p) = 0, \qquad \textit{(primal feasibility)} \tag{3b}$$
$$h(x,p) \geq 0, \tag{3c}$$
$$\mu \geq 0, \qquad \textit{(dual feasibility)} \tag{3d}$$
$$\mu^\top h(x,p) = 0, \qquad \textit{(complementarity slackness)} \tag{3e}$$

where (3e) *is equivalent to* $\mu_k h_k(x,p) = 0$ *for* $k = 1,\dots,n_h$. *Assume both the objective* $f(x,p)$ *and the constraints* $g(x,p)$, $h(x,p)$ *to be* $\mathscr{C}^1$ *with respect to* x *and Linear Independence Constraint Qualification (LICQ) to hold at* $x^*(p)$. *Then there exist Lagrange multipliers* $\lambda^*(p)$, $\mu^*(p)$ *such that* (3) *are satisfied at* $(x^*(p),\lambda^*(p),\mu^*(p))$.

Collecting primal and dual variables in a single vector $y := \begin{bmatrix} x^\top & \lambda^\top & \mu^\top \end{bmatrix}^\top$, we are interested in determining how the solution $y^*(p)$ changes as a result of small variations of p. The following theorem sets the basis for this analysis.

Theorem 2 (Optimal Sensitivity). *Suppose both the objective* $f(x,p)$ *and the constraints* $g(x,p)$, $h(x,p)$ *to be* $\mathscr{C}^2$ *with respect to* x *and* p. *For* $p \equiv \bar{p}$, *if FONC, Second Order Sufficient Conditions (SOSC), and Strict Complementarity Slackness (SCS) hold at* $y^*(\bar{p})$, *then: i)* $x^*(\bar{p})$ *is an isolated local minimum of* (mpNLP) *for* $p \equiv \bar{p}$ *and multipliers* $\lambda^*(\bar{p})$, $\mu^*(\bar{p})$ *are uniquely determined; ii) for* p *in a neighborhood of* $\bar{p}$, *there exists a unique* $\mathscr{C}^1$ *function* $y^*(p)$ *satisfying FONC, SOSC, and SCS for* (mpNLP).

Given the existence and the differentiability of $y^*(p)$, we can derive the sensitivity matrix $\mathrm{d}_p y^*(\bar{p})$. Let us consider the equalities (3a), (3b), (3e) of the FONC; requiring their total derivative with respect to p to vanish at the solution, we obtain the linear system of equations

$$\begin{bmatrix} \partial_{xx} l & -\partial_x^\top g & -\partial_x^\top h \\ \partial_x g & 0 & 0 \\ \mu^{*\top} \partial_x h & 0 & h^\top \end{bmatrix} \mathrm{d}_p y^* = - \begin{bmatrix} \partial_{xp} l \\ \partial_p g \\ \mu^{*\top} \partial_p h \end{bmatrix}, \tag{4}$$

where all the functions are evaluated at $\bar{p}$ and $y^*(\bar{p})$. Under the hypotheses of Theorem 2, the KKT matrix in the left hand side of (4) is always invertible and the linear system can be solved for $\mathrm{d}_p y^*(\bar{p})$.

Given a solution of (mpNLP) for some $p \equiv \bar{p}$, we are now able to approximate the solution of (mpNLP) for a new value of the parameter $p \equiv \bar{p} + \Delta_p$ as

$$y^*(\bar{p} + \Delta_p) \approx y^*(\bar{p}) + \mathrm{d}_p y^*(\bar{p}) \Delta_p. \tag{5}$$

4 Active Set Changes in Sensitivity Analysis

While the quality of approximation (5) is generally satisfactory when the perturbed parameters belong to a neighborhood of their nominal values $\bar{p}$, we assist to a severe accuracy loss whenever the magnitude of Δ_p is high enough to cause a change in the optimal active set (i.e., $\mathscr{A}^*(\bar{p}) \neq \mathscr{A}^*(\bar{p}+\Delta_p)$). This change, in fact, is associated with: i) a discontinuity in the sensitivity function $\mathrm{d}_p y^*(p)$, that makes the use of a linear approximation inconsistent; ii) a high risk of unfeasibility of the approximated solution which would result in a control input inapplicable to the real system.

In order to overcome this problem, in Sec. 4.1 we take advantage of a procedure (originally proposed in [8] and, more recently, used in NMPC [6]) that, through the definition of a tangent Quadratic Program (tQP), naturally extends the sensitivity analysis to take into account active set changes and inequality constraints (3c), (3d). In Sec. 4.2 we show how the structure of this tQP can be exploited to reduce offline the dimension of the problem, with a great decrease in the online solution times. Finally, in Sec. 4.3, we address the issue of the nonconvexity of the tQP and we propose a tailored convexification strategy which has a minimal impact on the solution. This leads to the statement of the problem we actually solve online.

4.1 Sensitivity Analysis via Quadratic Programming

In order to include active set changes in the sensitivity analysis, we consider the tQP

$$\begin{aligned} \min_{\Delta_x} \quad & \frac{1}{2}\Delta_x^\top \partial_{xx} l \Delta_x + \left(\Delta_p^\top \partial_{px} l + \partial_x f\right)\Delta_x \\ \text{s.t.} \quad & \left\{\partial_x g \Delta_x + \partial_p g \Delta_p = 0,\ h + \partial_x h \Delta_x + \partial_p h \Delta_p \geq 0\right\}, \end{aligned} \tag{tQP}$$

where all the functions are evaluated at $\bar{p}$ and $y^*(\bar{p})$ and Δ_x represents the correction to $x^*(\bar{p})$ induced by the parameter variation Δ_p. In this subsection we show that the solution of (tQP):

- always verifies the first order approximation at $\bar{p}$ and $y^*(\bar{p})$ of the feasibility conditions (3c) and (3d),
- coincides with the sensitivity approximation (5) whenever the latter does not violate the first order approximation of (3c) or (3d).

These two properties candidate (tQP) as the natural extention of (5) whenever active set changes are detected.

We start considering the Lagrangian function for (tQP) with multipliers $\lambda^*(\bar{p}) + \Delta_\lambda$ for the equalities and $\mu^*(\bar{p}) + \Delta_\mu$ for the inequalities. The related FONC are

$$\partial_{xx} l \Delta_x - \partial_x^\top g \Delta_\lambda - \partial_x^\top h \Delta_\mu + \partial_{xp} l \Delta_p = 0, \quad \text{(stationarity)} \tag{6a}$$

$$\partial_x g \Delta_x + \partial_p g \Delta_p = 0, \quad \text{(primal feasibility)} \tag{6b}$$

$$h + \partial_x h \Delta_x + \partial_p h \Delta_p \geq 0, \tag{6c}$$

$$\mu^* + \Delta_\mu \geq 0, \quad \text{(dual feasibility)} \tag{6d}$$

$$(\mu_k^* + \Delta_{\mu_k})(h_k + \partial_x h_k \Delta_x + \partial_p h_k \Delta_p) = 0, \quad \text{(complementarity slackness)} \tag{6e}$$

with $k = 1, \ldots, n_h$. Analyzing this set of equations we note that: (6a) and (6b) coincide with the first two equations in (4) multiplied by Δ_p, (6c) and (6d) are the linearization of (3c) and (3d) at $\bar{p}$ and $y^*(\bar{p})$, and (6e) is equivalent to the third equation in (4) except for second order terms. Focusing on (6e) we also notice that, as long as the solution of (6) verifies (3c) and (3d) strictly, second order terms are irrelevant and the solutions of (6) and (4) coincide. This claim can be easily verified; in fact, only one of the following alternatives is possible: either the kth constraint is active ($h_k > 0$, $\mu_k^* = 0$), then, since (6c) holds strictly, (6e) implies $\Delta\mu_k = 0$, which agrees with the result from the third equation in (4); or the kth constraint is inactive ($h_k = 0$, $\mu_k^* > 0$), then, since (6d) holds strictly, (6e) implies $\partial_x h_k \Delta_x + \partial_p h_k \Delta_p = 0$, which agrees with the result from the third equation in (4). On the other hand, second order terms in (6e) become relevant when some of the inequalities (6c), (6d) are active, ensuring feasibility of (6).

4.2 Condensation of the Tangent Quadratic Program

Generally speaking, there are two main approaches to the solution of (tQP) [6]: solve the problem in the actual form, using a sparse QP solver; or use the equality constraints to remove dependent variables (process called condensing) and solve the resulting dense QP. Since in our case the entire condensing process can be performed offline, the second approach is certainly more convenient. tQPs are typically condensed exploiting the structure of the control problem: with the initial state as a parameter, a forward simulation of the system dynamics is performed to express state variables as functions of the inputs and remove them from the optimization. This procedure however is not easily extendible to our problem because of the generic nature of the parameters we consider. In this subsection we propose a method to condense (tQP) that does not require any hypothesis on the typology of parameters.

Denoting with $\mathscr{N}(M)$ and $\mathscr{R}(M)$ the null and range space of a matrix M and defining $c_\mathrm{a}^\top := \begin{bmatrix} g^\top & h_\mathrm{a}^\top \end{bmatrix}$, we consider: $Z_1 \in \mathbb{R}^{n_x \times (n_x - n_g - n_\mathrm{a})}$ orthogonal basis of $\mathscr{N}(\partial_x c_\mathrm{a})$, $Z_2 \in \mathbb{R}^{n_x \times n_\mathrm{a}}$ orthogonal complement to Z_1 to form a basis of $\mathscr{N}(\partial_x g)$, and $Z_3 \in \mathbb{R}^{n_x \times n_g}$ orthogonal complement to $\begin{bmatrix} Z_1 & Z_2 \end{bmatrix}$ to form a basis of $\mathbb{R}^{n_x}$. Furthermore, we define $\tilde{Z}_2 := Z_2(\partial_x h_\mathrm{a} Z_2)^{-1}$, where $\partial_x h_\mathrm{a} Z_2 \in \mathbb{R}^{n_\mathrm{a} \times n_\mathrm{a}}$ is invertible since $\mathscr{N}(\partial_x h_\mathrm{a}) \cap \mathscr{R}(Z_2) = \emptyset$. We then consider the new set of variables $\begin{bmatrix} z_1^\top & \tilde{z}_2^\top & z_3^\top \end{bmatrix}^\top := \begin{bmatrix} Z_1 & \tilde{Z}_2 & Z_3 \end{bmatrix}^{-1} \Delta_x$. In the following we show that, with this change of variables, (tQP) assumes the condensed form

$$\min_{\tilde{z}_{12}} \frac{1}{2} \tilde{z}_{12}^\top H \tilde{z}_{12} + d\tilde{z}_{12} \text{ s.t. } \{A\tilde{z}_{12} \geq b,\ \tilde{z}_2 \geq c\}. \tag{ctQP}$$

where $\tilde{z}_{12}^\top := \begin{bmatrix} z_1^\top & \tilde{z}_2^\top \end{bmatrix}$. This reformulation entails two main advantages:

- the decrease in the number of optimization variables from n_x to $n_x - n_g$,
- the transformation of active inequalities into lower bounds on the variables $\tilde{z}_2$.

Changing variables in the equalities of (tQP), we can express the dependent variables as explicit functions of the parameter variation $z_3 = -(\partial_x g Z_3)^{-1} \partial_p g \Delta_p$, where again $\partial_x g Z_3 \in \mathbb{R}^{n_g \times n_g}$ is invertible since $\mathscr{N}(\partial_x g) \cap \mathscr{R}(Z_3) = \emptyset$. Active inequalities of (tQP) are transformed into lower bounds $\tilde{z}_2 \geq -\partial_x h_\mathrm{a} Z_3 z_3 - \partial_p h_\mathrm{a} \Delta_p =: c$. Defining $\tilde{Z}_{12} := \begin{bmatrix} Z_1 & \tilde{Z}_2 \end{bmatrix}$ and $A := \partial_x h_\mathrm{i} \tilde{Z}_{12}$, we see that inactive inequalities do not take any advantage from this change of variables, in fact we obtain $A\tilde{z}_{12} \geq -\partial_x h_\mathrm{i} Z_3 z_3 - \partial_p h_\mathrm{i} \Delta_p - h_\mathrm{i} =: b$. Finally, changing variables in the cost function, after some manipulations we arrive to the form presented in (ctQP) with $H := \tilde{Z}_{12}^\top \partial_{xx} l \tilde{Z}_{12}$ and $d := z_3^\top Z_3^\top \partial_{xx} l \tilde{Z}_{12} + \Delta_p^\top \partial_{px} l \tilde{Z}_{12} + \begin{bmatrix} 0_{1 \times (n_x - n_g - n_\mathrm{a})} & \mu_\mathrm{a}^{*\top} \partial_x h_\mathrm{a} \tilde{Z}_2 \end{bmatrix}$, where optimality condition (3a) have been used, constant terms (including z_3) have been removed, and $\mu_\mathrm{a}^* \in \mathbb{R}^{n_\mathrm{a}}$ denotes the optimal value of the active inequality multipliers.

4.3 Tangent Quadratic Program Convexification

The final issue in the solution of (tQP) is its potential nonconvexity, which prevents a direct application of the most efficient optimization algorithms. In fact, assuming SOSC and LICQ to hold, only the reduced Hessian matrix $R := Z_1^\top \partial_{xx} l Z_1$ is guaranteed to be positive definite, while $\partial_{xx} l$ might have negative eigenvalues.

Commonly, $\partial_{xx} l$ is convexified with the addition of a correction matrix $\Delta_L := \partial_x^\top c_\mathrm{a} \Lambda \partial_x c_\mathrm{a}$ with symmetric Λ. This choice is particularly appealing since:

- there always exists a finite Λ such that $\partial_{xx} l + \Delta_L \succ 0$,
- this convexification does not alter the solution of (tQP) whenever the correction from (tQP) does not require any active constraint to become inactive (i.e., whenever in (6c) we have $\partial_x h_\mathrm{a} \Delta_x^* + \partial_p h_\mathrm{a} \Delta_p = 0$, with Δ_x^* solution of (tQP)).

However, since we are going to solve the problem in its condensed form (ctQP), to our purpose is sufficient to make only H positive definite and not the entire $\partial_{xx} l$. To do that we consider the simpler correction matrix $\Delta_L := \partial_x^\top h_\mathrm{a} \Lambda \partial_x h_\mathrm{a}$; in the following we prove that the previous claims still hold.

We define the corrected Hessian matrix $L := \partial_{xx} l + \Delta_L$ and we replace it to $\partial_{xx} l$ in the definition of H. We get

$$\tilde{Z}_{12}^\top L \tilde{Z}_{12} = \begin{bmatrix} R & Z_1^\top \partial_{xx} l \tilde{Z}_2 \\ \tilde{Z}_2^\top \partial_{xx} l Z_1 & \tilde{Z}_2^\top \partial_{xx} l \tilde{Z}_2 + \Lambda \end{bmatrix}.$$

For the Schur complement lemma, requiring $\tilde{Z}_{12}^\top L \tilde{Z}_{12}$ to be positive definite is equivalent to require $R \succ 0$ and $\Lambda \succ \tilde{Z}_2^\top \left(\partial_{xx} l Z_1 R^{-1} Z_1^\top \partial_{xx} l - \partial_{xx} l \right) \tilde{Z}_2 =: S$. Being R the reduced Hessian, the first condition is verified whenever SOSC and LICQ hold, and being S a bounded matrix, a Λ such that the second relation holds can always be found. Moving to the second claim, we note that, as long as $\partial_x h_\mathrm{a} \Delta_x^* + \partial_p h_\mathrm{a} \Delta_p = 0$

the contribution to the cost function of the correction term $\Delta_x^{*\top} \Delta_L \Delta_x^*$ assumes the constant value $\Delta_p^\top \partial_p^\top h_\mathrm{a} \Lambda \partial_p h_\mathrm{a} \Delta_p$, hence it does not affect the solution of (tQP). Finally, concerning the choice of Λ, we decide to minimize $\|\Lambda\|_\mathrm{F}$ subject to $\min(\mathrm{eig}(\Lambda - S)) > \varepsilon$ for a small $\varepsilon > 0$, whose closed-form solution consists in a clipping of the eigenvalues of $\Lambda - S$ to ε [14].

4.4 Sensitivity Analysis Summary

In conclusion we briefly summarize the results from this section. We have shown how to take into account the first order approximation of inequality constraints (3c), (3d) in the sensitivity analysis. This process results in the quadratic program (tQP), whose solution coincides with the linear sensitivity correction (5) as long as the feasibility conditions (6c), (6d) are satisfied, but allows to improve the quality of the approximation for parameter variations Δ_p big enough to change the optimal active set. We have proposed a condensing strategy that (without any hypothesis on the nature of the parameters p) allows to remove (offline) all the dependent variables from (tQP), resulting in the dense, but substantially smaller, quadratic program (ctQP). Finally, we have proposed a convexification method tailored for (ctQP) which does not alter its solution as long as "constraint deactivations" do not occur, ensuring that also the solution of the convexified (ctQP) coincides with the linear correction (5) when the active set is unchanged.

5 Parametric Trajectory Libraries

We now describe how parametric trajectory libraries are built and how they are used for the online generation of suboptimal trajectories.

Having parameterized the OCP with the vector p, we discretize the continuous time problem into the finite dimensional (mpNLP). Alg. 1 is then applied for the construction of the trajectory library. This consists in the generation of random OCPs; for each OCP we first find the nearest trajectory in the library (using a metric defined in the following). Then, starting from this trajectory, an approximated solution is derived. The approximation accuracy is tested through a problem-dependent accuracy index: if the accuracy is high enough, the solution of (mpNLP) is not required, otherwise (mpNLP) is solved and its solution is added (together with sensitivity information) to the library. The algorithm stops when the number of consecutive successful approximations n_succ exceeds the threshold value $\underline{n}_\mathrm{succ}$.

Being the elements of p generally non-homogeneous, one of the key aspects of Alg. 1 is the definition of a metric in the parameter space. Our proposal is the following: when the kth trajectory is added to the library (with $k = 1, 2, \ldots, n_\mathrm{t}$), we compute $s_j^k := \|\mathrm{d}_{p_j} u^*(p^k)\|$ for $j = 1, 2, \ldots, n_p$. These values indicate how quickly the solution changes in the jth direction of the parameter space. We then perform

Algorithm 1: Parametric Trajectory Library Generation

```
Result: library
n_succ = 0
while n_succ < n̲_succ do
    generate random p
    if library is empty then
        solve (mpNLP)
        compute sensitivity (4) and (ctQP) blocks
        add trajectory to the library
    else
        find nearest trajectory in the library
        approximate solution with sensitivity (5)
        if constraints (6c) or (6d) are not verified then
            approximate solution with (ctQP)
        end
        if accuracy is poor then
            n_succ = 0
            solve (mpNLP)
            compute sensitivity (4) and (ctQP) blocks
            add trajectory to the library
        else
            n_succ = n_succ + 1
        end
    end
end
```

the search after scaling the jth parameter axis of the quantity $\frac{1}{n_t}\sum_{k=1}^{n_t} s_j^k$, so that directions with respect to which the solution is very sensitive on average are stretched, encouraging the choice of solutions in the shrunk directions. A minimization of the 2-norm distance is then used for the search in the scaled space.

Once the library is built, the online application of the method consists in the approximation steps of Alg. 1 exclusively: the parameter vector is assembled from sensor readings or external inputs; the nearest plan is selected; the approximation from the sensitivity analysis is accepted if the linearized feasibility constraints are verified, otherwise the solution of the (ctQP) is required. The approximation is then applied to the system without any further verification.

5.1 Application to the Benchmark Problem

In Sec. 2 we have shown that elastic actuators can in principle reduce the energy consumption required to perform a task, but standard trajectory optimization techniques are not sufficiently fast to exploit the robot elasticity. Using trajectory libraries, on the other hand, the online computation cost is reduced to the solution of a linear system and, when needed, a QP. Here we present the results obtained using a parametric trajectory library for the solution of the benchmark problem of Sec. 2.

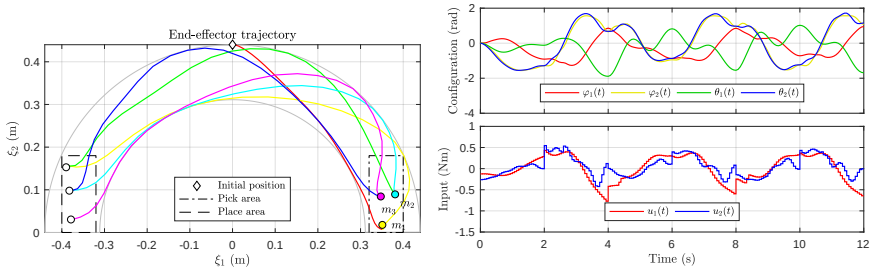


Fig. 3: First three pick-and-place tasks for the constant-stiffness robot performed applying the approximated optimal planning from the parametric trajectory library.

(OCP) is parameterized with the initial and final configuration of the robot and the mass m_o of the object that the manipulator has to move. Considering that final motor positions $\theta(t_\mathrm{f}^k)$ are free, and exploiting the circular symmetry of the problem, the number of parameters is $n_p = 6$. Describing the end-effector position with polar coordinates $\rho := \|\xi\|$ and $\alpha := \mathrm{atan2}(\xi_2, \xi_1)$, the parameter vector is $p^k := \left[\alpha_\mathrm{f}^k - q_{\mathrm{i}_1}^k \; q_{\mathrm{i}_2}^k \; \theta_{\mathrm{i}_1}^k \; \theta_{\mathrm{i}_2}^k \; \rho_\mathrm{f}^k \; m^k\right]^\top \in \mathbb{R}^{n_p}$, with $k = 1, 2, \ldots, 2n_\mathrm{o}$. An additional symmetry can be exploited noticing that two parameter vectors p^k and $p^j \equiv \left[-p_1^k \; -p_2^k \; -p_3^k \; -p_4^k \; p_5^k \; p_6^k\right]^\top$ are associated to solutions with opposite sign. Random plannings for the construction of the library are generated with the procedure presented in Sec. 2. In order to test the accuracy of the approximated optimal control law we generate, we use a grasp error index defined as $e_\mathrm{g}^k := \|\xi(t_\mathrm{f}^k) - \xi_\mathrm{f}^k\| + t_\mathrm{g}\|\dot{\xi}(t_\mathrm{f}^k)\|$, with t_g denoting the hand closure time and $\xi(t_\mathrm{f}^k)$ the simulated final position of the end-effector. This represents an upper bound of the distance between the object and the end-effector during the interval necessary to close the gripper. The threshold value for the grasp error is set to 10 mm.

After 47315 random plannings the algorithm stops, succeeding in $\underline{n}_\mathrm{succ} = 10^3$ consecutive approximations. The number of trajectories added to the library is $n_\mathrm{t} = 578$, which is considerably small considering that $n_\mathrm{t}^{1/n_p} < 3$. Solving the sequence of $2n_\mathrm{o}$ plannings analyzed in Sec. 2 relying on approximated solutions exclusively, we obtain an average cost of $\bar{J}_\mathrm{SC}^* = 0.137\ (\mathrm{Nm})^2$, which is identical to the non-approximated case. The average grasp error is 1.92 mm, with a maximum value of 10.78 mm. 22.5% of the times sensitivity correction from (4) is sufficient, whereas in the other cases the solution of (ctQP) is required. Simulations of the first six approximated optimal controls are illustrated in Fig. 3, where the trajectory of the end-effector reveals some small positioning errors. Using the QP solver `quadprog` from `MATLAB`, the solution of (ctQP) requires on average 46.0 ms (with a maximum value of 79.1 ms). In conclusion we note that repeating this analysis eliminating the whole correction process (i.e., using the nearest trajectory in the library directly) we obtain an average grasp error of 32.2 mm with a maximum value of 88.7 mm; errors that are approximately ten times bigger that the ones obtained after the refinement.

6 Experimental Validation

In this section the method of parametric trajectory libraries is experimentally validated on the manipulator shown in Fig. 1; this is powered by series elastic actuators, and provided with the fast version of the Pisa-IIT SoftHand [3]. Stiffness values of the series elastic actuators are not optimal but equal to 0.7 Nm/rad; nonetheless, the goal of these experiments is to show the online applicability of the proposed method. A detailed analysis of energy consumptions will be subject of future works. Experimental results are also presented in the video accompanying the paper[1].

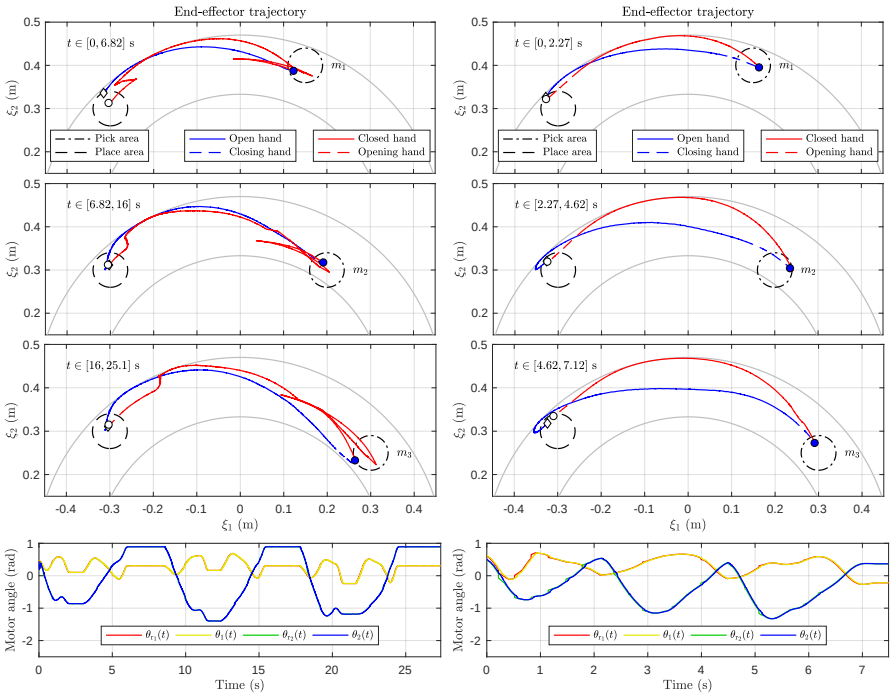
Motor positions are controlled with a PID controller, tracking the reference signal $\theta_\mathrm{r}(t)$ generated by the optimization process (note that this closure of the loop does not alter the elastic properties of the system). The task is to pick three objects with mass 100 g from the positions $\xi_\mathrm{f}^1 = \begin{bmatrix}15 & 40\end{bmatrix}^\top$ cm, $\xi_\mathrm{f}^3 = \begin{bmatrix}20 & 30\end{bmatrix}^\top$ cm, $\xi_\mathrm{f}^5 = \begin{bmatrix}30 & 25\end{bmatrix}^\top$ cm, and to place them in $\xi_\mathrm{f}^2 = \xi_\mathrm{f}^4 = \xi_\mathrm{f}^6 = \begin{bmatrix}-30 & 30\end{bmatrix}^\top$ cm. The time horizon for each movement is 1.5 s. The closure (opening) of the hand starts when the condition $\frac{1}{2}\mathrm{d}_t\|\xi(q)-\xi_\mathrm{f}^k\|t_\mathrm{g} > \|\xi(q)-\xi_\mathrm{f}^k\|$ is verified, with $k = 1,\dots,6$.

We start presenting in Fig. 4a the results achievable when each movement is planned through the solution of (mpNLP). The first thing to note is that, because of the high computation times, in this case a settling time is necessary to stabilize the system at the end of each planning, ensuring that the position read by the sensors (and used as initial condition in (OCP)) is consistent with the state of the system when the feedforward is actually applied. (Settling times can be easily recognized from Fig. 4a as time intervals of constant motor angle.) This phenomenon makes the storage of potential energy between consecutive tasks impossible therefore, in order to facilitate the stabilization, we add the constraint $\varphi(t_\mathrm{f}^k) = \theta(t_\mathrm{f}^k)$ to (OCP). Considering the absence of a feedback action on the end-effector position, the overall accuracy of the movements is satisfactory (even if not sufficient to guarantee the success of every grasp): each trajectory reaches a circle of radius 40 mm centered at the nominal final position. Nonetheless, stabilization phases make the system behavior and the execution time (25.1 s) unacceptable. Fig. 4b shows the experiment in case a trajectory library is used. Here it can be seen that, with similar accuracy, the method based on the trajectory libraries is able to perform the same task in a considerably shorter time (7.1 s) and without requiring any settling time.

7 Conclusions and Future Works

In this paper we have illustrated a procedure for online generation of optimal motion plans for constrained nonlinear systems. Combining a trajectory library with a sensitivity analysis, we are able to refine online the trajectories stored in the library, making them suitable for the open-loop control of the system. Analyzing this problem as a generic parametric NLP, we can manage a wider class of parameters with

[1] The accompanying video can be found at https://www.youtube.com/watch?v=AEDXAZmoPuw.



(a) Nonlinear trajectory optimization. (b) Parametric trajectory library.

Fig. 4: Experimental results. From top to bottom: end-effector trajectory during the three pick-and-place actions, reference and real motor positions with the PID controller.

respect to common applications of trajectory libraries, which are mainly focused on system stabilization. We motivated the necessity of such a planning technique analyzing a pick-and-place task, showing how the combination of soft actuation and trajectory optimization can drastically reduce energy consumptions but it is not feasible because of the slow responsiveness of NLP-based planners. In this context, we have experimentally validated our approach demonstrating its ability to overcome computation time issues for the analyzed system. Nonetheless, we underline that the presented method can be applied to the trajectory optimization of generic dynamical systems and is not in any way restricted to soft robots or robots in general.

Future works will be focused on the analysis of the complexity of the presented approach in case of higher dimensional systems, such as bipeds and humanoids. Moreover, we plan to conduct a deeper analysis of the sampling process for the library generation in order to derive theoretical guarantees about its completeness. Finally, we also aim to conduct more detailed experimental analyses on the energy benefits that result from the application of this method to soft robots.

References

1. Andersson, J.: A General-Purpose Software Framework for Dynamic Optimization. PhD thesis, Arenberg Doctoral School, KU Leuven (2013)
2. Betts, J.T.: Practical methods for optimal control and estimation using nonlinear programming, vol. 19. Siam (2010)
3. Catalano, M.G., Grioli, G., Farnioli, E., Serio, A., Piazza, C., Bicchi, A.: Adaptive synergies for the design and control of the pisa/iit softhand. International Journal of Robotics Research **33**, 768–782 (2014)
4. Della Santina, C., Bianchi, M., Grioli, G., Angelini, F., Catalano, M., Garabini, M., Bicchi, A.: Controlling soft robots. IEEE Robotics & Automation Magazine (2017)
5. Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., Allgöwer, F.: Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. Journal of Process Control **12**(4), 577–585 (2002)
6. Diehl, M., Gros, S.: Numerical optimization of dynamic systems (draft) (2016). URL https://www.syscop.de/files/2015ws/noc-dae/00-book.pdf
7. Fiacco, A.V.: Introduction to sensitivity and stability analysis in nonlinear programming. Academic Press, Inc., by Springer-Verlag (1984)
8. Ganesh, N., Biegler, L.T.: A reduced hessian strategy for sensitivity analysis of optimal flowsheets. AIChE Journal **33**(2), 282–296 (1987)
9. Garabini, M., Passaglia, A., Belo, F., Salaris, P., Bicchi, A.: Optimality principles in variable stiffness control: The vsa hammer. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 3770–3775. IEEE (2011)
10. Haddadin, S., Huber, F., Albu-Schäffer, A.: Optimal control for exploiting the natural dynamics of variable stiffness robots. In: Robotics and Automation (ICRA), IEEE International Conference on, pp. 3347–3354. IEEE (2012)
11. Kim, S., Laschi, C., Trimmer, B.: Soft robotics: a bioinspired evolution in robotics. Trends in biotechnology **31**(5), 287–294 (2013)
12. Liu, C., Atkeson, C.G.: Standing balance control using a trajectory library. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3031–3036. IEEE (2009)
13. Mordatch, I., Todorov, E., Popović, Z.: Discovery of complex behaviors through contact-invariant optimization. ACM Transactions on Graphics (TOG) **31**(4), 43 (2012)
14. Nocedal, J., Wright, S.: Numerical optimization. Springer Science & Business Media (2006)
15. Schaal, S., Atkeson, C.G.: Learning control in robotics. IEEE Robotics & Automation Magazine **17**(2), 20–29 (2010)
16. Silva, M.F., Machado, J.T.: A literature review on the optimization of legged robots. Journal of Vibration and Control **18**(12), 1753–1767 (2012)
17. Stolle, M., Atkeson, C.G.: Policies based on trajectory libraries. In: Robotics and Automation (ICRA), IEEE International Conference on, pp. 3344–3349 (2006)
18. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, J.W.: Lqr-trees: Feedback motion planning via sums-of-squares verification. The International Journal of Robotics Research (2010)
19. Vanderborght, B., Albu-Schäffer, A., Bicchi, A., Burdet, E., Caldwell, D.G., Carloni, R., Catalano, M., Eiberger, O., Friedl, W., Ganesh, G., et al.: Variable impedance actuators: A review. Robotics and Autonomous Systems **61**(12), 1601–1614 (2013)
20. Vanderborght, B., Verrelst, B., Van Ham, R., Van Damme, M., Lefeber, D., Duran, B.M.Y., Beyl, P.: Exploiting natural dynamics to reduce energy consumption by controlling the compliance of soft actuators. The International Journal of Robotics Research (2006)
21. Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J., Diehl, M.: Practical time-optimal trajectory planning for robots: a convex optimization approach. IEEE Transactions on Automatic Control (2008)
22. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming (2006)
23. Zavala, V.M., Biegler, L.T.: The advanced-step nmpc controller: Optimality, stability and robustness. Automatica **45**(1), 86–93 (2009)