Robotic Arm Weightlifting via Trajectory Optimization

Final Project Report for MIT 6.843: Robotic Manipulation, Fall 2021

Timur Garipov
MIT CSAIL

***Abstract*—In this work we build a robotic arm controller in Drake that executes weightlifting motion. Our solution is based on trajectory optimization approach. We formulate the weightlifting task as a trajectory optimization problem and numerically solve for the optimal trajectory using direct collocation and nonlinear optimization. We evaluate the proposed solution in simulation across various weight configurations and compare against a manually-designed baseline trajectory. A special care is given to handling of the torque limit constraints which play a key role in the weightlifting task.**

I. Introduction

Many manipulations tasks require execution of complex coordinated motions. This work focuses on a particular manipulation task: robotic arm weightlifting. The goal of the weightlifting task is to find an actuation signal for a robotic arm which executes a motion resulting in lifting of a heavy object attached to the arm. The dynamics of a robotic arm with an attached object is governed by nonlinear ordinary differential equations. Finding an actuation signal for the successful execution of the motion is nontrivial since the robot arm motors must counteract the gravity with limited effort and ensure a steady lifting motion.

We apply trajectory optimization approach for the weightlifting task and evaluate obtained trajectories in simulation in Drake [1]. Our approach is based on direct collocation method [2] for trajectory optimization. The optimal trajectory is found numerically as a solution to a nonlinear optimization problem. We empirically evaluate optimized trajectories in simulation and demonstrate that the optimization method can find trajectories which execute valid lifting motions across different weight configurations.

II. Problem Statement

Our goal is to lift a heavy object with a robotic arm. We assume that, the arm base is stationary, the object is rigid, and the pose of the object relative to the last link of the arm is fixed. The configuration of the arm and the object is described by the position vector (vector of joint angles) $\boldsymbol{q} \in \mathbb{R}^d$ of the arm, where d is the number of joints in the arm (i.e. the number of DOFs). Let O be a point associated with the object. We denote the position of the object expressed in the world frame by p^O. Under our assumptions the pose of the object is fully determined by the position $\boldsymbol{q}$ of the arm and the position of object $p^O(\boldsymbol{q})$ can be computed using the forward kinematics map of the arm. The height to which the object is lifted in a given configuration $\boldsymbol{q}$ can be computed as the z-coordinate of O in the world frame: $\left[p^O\right]_z(\boldsymbol{q})$. Below, we will use a shorthand notation $z^O(\boldsymbol{q}) = \left[p^O\right]_z(\boldsymbol{q})$.

We consider a fully-actuated robotic arm with actuation input $\boldsymbol{u} \in \mathbb{R}^d$ representing the vector of torques applied at each of the arm's joints. Given an initial position $\boldsymbol{q}_{\text{start}}$, initial velocities $\dot{\boldsymbol{q}}_{\text{start}}$ and an input signal $\boldsymbol{u}(t)$ for $t \in [0, T]$ the system evolves according to the ODE with the initial condition

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad t \in [0, T], \tag{1}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_{\text{start}}, \tag{2}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{2d}$ is the dynamical state of the system

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix}, \qquad \boldsymbol{x}_{\text{start}} = \begin{bmatrix} \boldsymbol{q}_{\text{start}} \\ \dot{\boldsymbol{q}}_{\text{start}} \end{bmatrix}, \tag{3}$$

and function $f : \mathbb{R}^{2d} \times \mathbb{R}^d \to \mathbb{R}^{2d}$ defines the state equation (1) derived from the Newton's laws of motion expressed in generalized coordinates $\boldsymbol{q}$.

In the setup described above we formulate the desiderata for the solution of the weightlifting task. Given an initial state $\boldsymbol{x}_{\text{start}}$ we seek an actuation input signal $\boldsymbol{u}(t), t \in [0, T]$ with the following properties.

- **Maximal lift height**. Following the actuation signal $\boldsymbol{u}(t)$ the system evolves according to (1)-(2) from the initial state $\boldsymbol{x}(0) = \boldsymbol{x}_{\text{start}} = [\boldsymbol{q}_{\text{start}}, \dot{\boldsymbol{q}}_{\text{start}}]$ to the final state $\boldsymbol{x}(T) = [\boldsymbol{q}(T), \dot{\boldsymbol{q}}(T)]$ so that the lift height $z^O(\boldsymbol{q}(T))$ is as high as possible.
- **Torque limits constraint**. We require that the actuation signal respects the torque limits of the arm:

$$-\boldsymbol{u}_{\text{lim}} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\text{lim}}, \quad t \in [0, T]. \tag{4}$$

  The torque limit constraint plays an essential role in the weightlifting task. Indeed, all robot arm actuators can exert limited torques in practice and an input trajectory $\boldsymbol{u}(t)$ violating the torque limits cannot be executed on a real robot arm. Moreover, even if one works in a simulation and does not model all aspects of the real robot arm, without torque limits the weightlifting task becomes trivial as any arbitrarily heavy weight can be lifted by exerting unbounded torques at the arm's joints.

III. BASELINE: MANUALLY-DESIGNED TRAJECTORY

In this section we describe a simple manually-designed trajectory for weightlifting task. We use this simple solution as a baseline for our trajectory optimization based solution described in Section IV.

To construct a baseline trajectory $(\boldsymbol{u}_{\text{base}}(t), \boldsymbol{x}_{\text{base}}(t))$, we choose a duration T and define a piece-wise linear nominal position trajectory $\boldsymbol{q}^{\text{nom}}_{\text{base}}(t)$:

$$\boldsymbol{q}^{\text{nom}}_{\text{base}}(t) = \begin{cases} \frac{(0.9T - t)\cdot \boldsymbol{q}_{\text{start}} + t \cdot \boldsymbol{q}_{\text{upright}}}{0.9T}, & t \in [0, 0.9T], \\ \boldsymbol{q}_{\text{upright}}, & t \in [0.9T, T], \end{cases} \tag{5}$$

where $\boldsymbol{q}_{\text{upright}}$ corresponds to the upright position of the arm. Position trajectory $\boldsymbol{q}^{\text{nom}}_{\text{base}}(t)$ linearly interpolates between the starting position $\boldsymbol{q}_{\text{start}}$ and the upright position $\boldsymbol{q}_{\text{upright}}$ in the time segment $[0, 0.9T]$, and the holds the constant upright position $\boldsymbol{q}^{\text{nom}}_{\text{base}}(t) = \boldsymbol{q}_{\text{upright}}$ for $t \in [0.9T, T]$.

Having $\boldsymbol{q}^{\text{nom}}_{\text{base}}(t)$ we define the nominal baseline state trajectory $\boldsymbol{x}^{\text{nom}}_{\text{base}}(t) = [\boldsymbol{q}^{\text{nom}}_{\text{base}}(t), \dot{\boldsymbol{q}}^{\text{nom}}_{\text{base}}(t)]$ by simply computing the derivative $\dot{\boldsymbol{q}}^{\text{nom}}_{\text{base}}(t)$ of the position trajectory $\boldsymbol{q}^{\text{nom}}_{\text{base}}(t)$.

Finally, we compute the baseline $(\boldsymbol{u}_{\text{base}}(t), \boldsymbol{x}_{\text{base}}(t))$ by simulating the tracking of the nominal state trajectory $\boldsymbol{x}^{\text{nom}}_{\text{base}}(t)$ with the inverse dynamics controller (PID + inverse dynamics as implemented by `InverseDynamicsController` in Drake).

Note that in the baseline trajectory the actuation signal $\boldsymbol{u}_{\text{base}}(t)$ is not specified explicitly but rather computed by the controller. Lack of explicit control of the torques $\boldsymbol{u}_{\text{base}}(t)$ is a serious drawback of the baseline solution, in our experiments in Section V we observe that the baseline trajectory can successfully execute the weightlifting motion without violating the torque limits only for relatively light weights.

IV. WEIGHTLIFTING VIA TRAJECTORY OPTIMIZATION

A. Trajectory Optimization formulation

We cast the weightlifting task as a trajectory optimization problem:

$$\min_{T, \boldsymbol{u}(\cdot), \boldsymbol{x}(\cdot)} \quad \ell_f(\boldsymbol{x}(T)) + \int_0^T \ell_r(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt, \tag{6}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \qquad t \in [0, T], \tag{7}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_{\text{start}}, \tag{8}$$

$$T_{\text{min}} \le T \le T_{\text{max}}, \tag{9}$$

$$\boldsymbol{x}_{\text{min}} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{\text{max}}, \tag{10}$$

$$-\boldsymbol{u}_{\text{lim}} \le \boldsymbol{u}(t) \le \boldsymbol{u}_{\text{lim}}. \tag{11}$$

Given the initial state of the system $\boldsymbol{x}_{\text{start}} = [\boldsymbol{q}_{\text{start}}, \dot{\boldsymbol{q}}_{\text{start}}]$ we seek the optimal control $u(t)$ and the state trajectory $\boldsymbol{x}(t)$, $t \in [0, T]$ (note that the duration of the trajectory T is also a decision variable). The optimal trajectory $(\boldsymbol{u}_{\text{opt}}(\cdot), \boldsymbol{x}_{\text{opt}}(\cdot))$ incurs the minimum possible trajectory cost (6) while satisfying the constraints (7)-(11).

a) Cost function: The trajectory cost (6) is defined by the final cost function $\ell_f(\cdot)$ and the running cost function $\ell_r(\cdot)$.

For the weightlifting task, we specify the final cost function

$$\ell_f(\boldsymbol{x}(T)) = \ell_f\left(\begin{bmatrix}\boldsymbol{q}(T)\\ \dot{\boldsymbol{q}}(T)\end{bmatrix}\right) = -\lambda_{z,f} \cdot z^O(\boldsymbol{q}(T)) + \lambda_v \cdot \|\dot{\boldsymbol{q}}(T)\|^2. \tag{12}$$

The final cost function (12) is a weighted sum of two terms,

- $-z^O(\boldsymbol{q}(T))$ — negated height of object at the end of the trajectory with weight $\lambda_{z,f} > 0$;
- $\|\dot{\boldsymbol{q}}(T)]\|^2$ — squared norm of the final velocity with weight $\lambda_v > 0$.

With this final cost function the trajectory optimization maximizes possible final lift height and minimizes the final joint velocities. The final velocity cost is added to promote stability of the lifting motion: a solution where the arm lifts the weight and secures it in a stationary position ($\|\dot{\boldsymbol{q}}(T)]\|^2 \approx 0$) is preferred.

We define the running cost function as

$$\ell_r(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \ell_r\left(\begin{bmatrix}\boldsymbol{q}(t)\\ \dot{\boldsymbol{q}}(t)\end{bmatrix}, \boldsymbol{u}(t)\right) = -\lambda_{z,r} \cdot z^O(\boldsymbol{q}(t)) + \lambda_u \cdot \|\boldsymbol{u}(t)\|^2. \tag{13}$$

The running cost function (13) is a weighted combination of two terms:

- $-z^O(\boldsymbol{q}(t))$ — negated height of object at $t \in [0, T]$ with weight $\lambda_{z,r} > 0$;
- $\|u(t)\|^2$ — the squared norm of the torque vector $\boldsymbol{u}(t)$ with weight $\lambda_u > 0$.

With this running cost the trajectory optimization maximizes integral of the lift height and minimizes the integral of the magnitude of the torque signal $\boldsymbol{u}(t)$. The lift height term is included in the running cost (in addition to the final cost) in order to promote faster lifting motions (the earlier the object is lifted the smaller is the integral of the lift height cost). The torque magnitude term acts as a regularizer of the input signal and promotes the lifting motions which require smaller torques applied at the arm joints.

b) Constraints: Constraint (7) ensures that the trajectory $(\boldsymbol{u}(\cdot), \boldsymbol{x}(\cdot))$ follows a valid dynamics. Constraint (8) enforces the initial state condition. Constraint (9) specifies the lower and upper limits $(T_{\text{min}}, T_{\text{max}})$ on the trajectory duration T. Constraint (10) enforces position and velocity limits of the arm's joints. Finally, constraint (11) limits the torques applied at the arm joints.

B. Numerical Trajectory Optimization via Direct Collocation

We use direct collocation approach [2] to solve the trajectory optimization problem (6)-(11) numerically.

In direct collocation approach, an infinite-dimensional continuous-time optimization problem is transcribed into a finite-dimensional discrete-time mathematical program. After the discrete-time mathematical program is solved the approximation of the optimal trajectory $(\boldsymbol{u}(t), \boldsymbol{x}(t))$ is reconstructed from the discrete-time sample values $\boldsymbol{u}[\cdot], \boldsymbol{x}[\cdot]$:

$$\boldsymbol{u}[n] = \boldsymbol{u}(t[n]), \quad \boldsymbol{x}[n] = \boldsymbol{x}(t[n]), \quad 0 \le n \le N, \tag{14}$$
$$0 = t[0] < t[1] < \ldots < t[n-1] < t[N] = T, \tag{15}$$

where N is the number of time samples.

We use a particular direction collocation method where the reconstructed input trajectory $u(t)$ is represented by a piece-wise linear function

$$\boldsymbol{u}(t) \text{ is linear on } [t[n], t[n+1]], \tag{16}$$
$$\boldsymbol{u}(t[n]) = \boldsymbol{u}[n], \tag{17}$$
$$\boldsymbol{u}(t[n+1]) = \boldsymbol{u}[n+1], \tag{18}$$

parameterized by the sample values $\{\boldsymbol{u}[n]\}_{n=0}^{N}$ and the reconstructed state trajectory is represented by piece-wise cubic polynomial

$$\boldsymbol{x}(t) \text{ is cubic on } [t[n], t[n+1]], \tag{19}$$
$$\boldsymbol{x}(t[n]) = \boldsymbol{x}[n], \tag{20}$$
$$\boldsymbol{x}(t[n+1]) = \boldsymbol{x}[n+1], \tag{21}$$
$$\dot{\boldsymbol{x}}(t[n]) = f(\boldsymbol{x}[n], \boldsymbol{u}[n]), \tag{22}$$
$$\dot{\boldsymbol{x}}(t[n+1]) = f(\boldsymbol{x}[n+1], \boldsymbol{u}[n+1]), \tag{23}$$

parameterized by the sample values $\{\boldsymbol{x}[n]\}_{n=0}^{N}$, $\{\boldsymbol{u}[n]\}_{n=0}^{N}$.

With the above parameterization of the trajectories, the original trajectory optimization problem (6)-(11) can be transcribed into the following finite-dimensional mathematical program

$$\min_{h[\cdot], \boldsymbol{u}[\cdot], \boldsymbol{x}[\cdot]} \ \ell_f(\boldsymbol{x}[N]) + \sum_{n=0}^{N-1} h[n] \ell_r(\boldsymbol{x}[n], \boldsymbol{u}[n]), \tag{24}$$
$$\text{s.t.} \quad F(h[n], \boldsymbol{x}[n], \boldsymbol{x}[n+1], \boldsymbol{u}[n], \boldsymbol{u}[n+1]) = 0, \tag{25}$$
$$\boldsymbol{x}[0] = \boldsymbol{x}_{\text{start}}, \tag{26}$$
$$\frac{T_{\min}}{N} \le h[n] \le \frac{T_{\max}}{N}, \tag{27}$$
$$\boldsymbol{x}_{\min} \le \boldsymbol{x}[n] \le \boldsymbol{x}_{\max}, \tag{28}$$
$$-\boldsymbol{u}_{\text{lim}} \le \boldsymbol{u}[n] \le \boldsymbol{u}_{\text{lim}}, \tag{29}$$

where $h[\cdot]$ are the the timesteps

$$h[n] = t[n+1] - t[n], \quad 0 \le n \le N-1. \tag{30}$$

Equation (25) gives the dynamics constraints at the collocation points $t_c[n] = \frac{1}{2}(t[n] + t[n+1])$. The function F is defined as

$$F(h, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u}_1, \boldsymbol{u}_2) = \dot{\boldsymbol{x}}_c - f(\boldsymbol{x}_c, \boldsymbol{u}_c), \tag{31}$$

where the values $\boldsymbol{x}_c, \dot{\boldsymbol{x}}_c, \boldsymbol{u}_c$ depend on $h, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u}_1, \boldsymbol{u}_2$:

$$\dot{\boldsymbol{x}}_1 = f(\boldsymbol{x}_1, \boldsymbol{u}_1), \tag{32}$$
$$\dot{\boldsymbol{x}}_2 = f(\boldsymbol{x}_2, \boldsymbol{u}_2), \tag{33}$$
$$\boldsymbol{u}_c = \frac{1}{2}(\boldsymbol{u}_1 + \boldsymbol{u}_2), \tag{34}$$
$$\boldsymbol{x}_c = \frac{1}{2}(\boldsymbol{x}_1 + \boldsymbol{x}_2) + \frac{h}{8}(\dot{\boldsymbol{x}}_1 - \dot{\boldsymbol{x}}_2), \tag{35}$$
$$\dot{\boldsymbol{x}}_c = -\frac{3}{2h}(\boldsymbol{x}_1 - \boldsymbol{x}_2) - \frac{1}{4}(\dot{\boldsymbol{x}}_1 + \dot{\boldsymbol{x}}_2). \tag{36}$$

We use the mathematical programming module in Drake to solve the nonlinear program (24)-(29). Internally, for non-linear programs Drake uses SNOPT SQP solver [3], [4]. We use the baseline trajectory as $(\boldsymbol{u}_{\text{base}}(t), \boldsymbol{x}_{\text{base}}(t))$ described in Section III as an initial guess for the solver. After solving for the optimal sample values $h_{\text{opt}}[\cdot]$, $\boldsymbol{x}_{\text{opt}}[\cdot]$, $\boldsymbol{u}_{\text{opt}}[\cdot]$ we reconstruct the nominal optimal trajectory $(\boldsymbol{u}_{\text{opt}}^{\text{nom}}(t), \boldsymbol{x}_{\text{opt}}^{\text{nom}}(t))$ using (16)-(18) and (19)-(23).

C. Trajectory Tracking

The nominal optimal trajectory $(\boldsymbol{u}_{\text{opt}}^{\text{nom}}(t), \boldsymbol{x}_{\text{opt}}^{\text{nom}}(t))$ found as the numerical solution of the non-linear program (24)-(29) is not guaranteed to exactly follow the system dynamics (1)-(2).

- The mathematical program (24)-(29) has a large number of decision variables which grows linearly with the number of the time samples N. In order to limit the number of decision variables one has to use relatively coarse discrete-time steps which inevitably introduces discretization errors in the reconstructed trajectories.
- The constraints (25) are nonlinear. SNOPT's solution might violate nonlinear constraints (in this case SNOPT attempts to minimize the sum of violations).

In order to correct the errors in the nominal trajectory $(\boldsymbol{u}_{\text{opt}}^{\text{nom}}(t), \boldsymbol{x}_{\text{opt}}^{\text{nom}}(t))$, we apply trajectory tracking. Specifically, the final optimal trajectory $(\boldsymbol{u}_{\text{opt}}(t), \boldsymbol{x}_{\text{opt}}(t))$ is obtained as the result of tracking of the nominal state trajectory $\boldsymbol{x}_{\text{opt}}^{\text{nom}}(t)$ with the inverse dynamics controller (using Drake's `InverseDynamicsController`, similarly to tracking of the nominal state trajectory $\boldsymbol{x}_{\text{base}}^{\text{nom}}(t)$ in Section III). The resulting actuation signal $\boldsymbol{u}_{\text{opt}}(t)$ in general is different from the nominal control $\boldsymbol{u}_{\text{opt}}^{\text{nom}}(t)$. Therefore, we monitor the tracking trajectory $\boldsymbol{u}_{\text{opt}}(t)$ and check that the commanded torques respect the torque limits (4) throughout the entire duration of the trajectory. The

V. Results

We implemented and tested the baseline solution (Section III) and the trajectory optimization based solution (Section IV) in Drake [1]. We used KUKA LBR iiwa 7DOF robot arm. The parameters used in evaluation are described below:

- **Torque limits**. We used torque limits provided in the official technical data brochure for the KUKA LBR iiwa robot arm [5]:

$$\boldsymbol{u}_{\text{lim}} = [176\,\text{Nm}, 176\,\text{Nm}, 110\,\text{Nm}, 110\,\text{Nm}, 110\,\text{Nm}, 40\,\text{Nm}, 40\,\text{Nm}]. \tag{37}$$

- **Initial position**.

$$q_{\text{start}} = [0, 1.9, 0, -\pi + 1.9, 0, 0, 0]. \quad (38)$$

- **Upright position**.

$$q_{\text{upright}} = [0, 0, 0, 0, 0, 0, 0]. \quad (39)$$

- **Trajectory duration**. For the baseline trajectory (5) we set duration

$$T = 10\,s. \quad (40)$$

For direct collocation (24)-(29) we set the duration limits

$$T_{\min} = 5\,\text{s}, \quad T_{\max} = 10\,\text{s}. \quad (41)$$

The results of trajectory tracking for different weight configurations are shown in Figures 1, 2. We tested the following weight configurations.

- Ball weights with masses 5 kg, 15 kg, 17 kg (Figure 1 (a)-(c)).
- Balanced dumbbells (comprised of a cylinder bar and two ball weights) with total masses 10 kg and 19 kg (Figure 2 (a)-(b)).
- Imbalanced dumbbell of total mass 15 kg distributed over two balls $13.5\,\text{kg} + 1.5\,\text{kg}$ (Figure 2 (c)).

We observe that the baseline trajectory can be successfully executed without torque limits violation for moderately heavy weights: Figure 1 (a)-(b), Figure 2 (a). In these cases, the optimization method finds trajectories which execute the lifting motion in a shorter time and better utilize the effort (as measured by the torque norm). For heavier weights (Figure 1 (a)-(c), Figure 2 (b)-(c)), the baseline trajectory violates the torque limits, while the optimization procedure finds a valid signal respecting the constraints. In all cases, the inverse dynamics controller can accurately track the nominal state trajectory.

VI. Discussion and Future Work

In this work, we implemented a robotic arm controller in Drake which executes weightlifting motion. Our results demonstrate the effectiveness of trajectory optimization approach for the weightlifting task. The potential directions for future work include further development of the system to support

- grasping of the object with a gripper and modeling of friction forces between the gripper and the object during lifting motion;
- additional constraints such as collision avoidance.

References

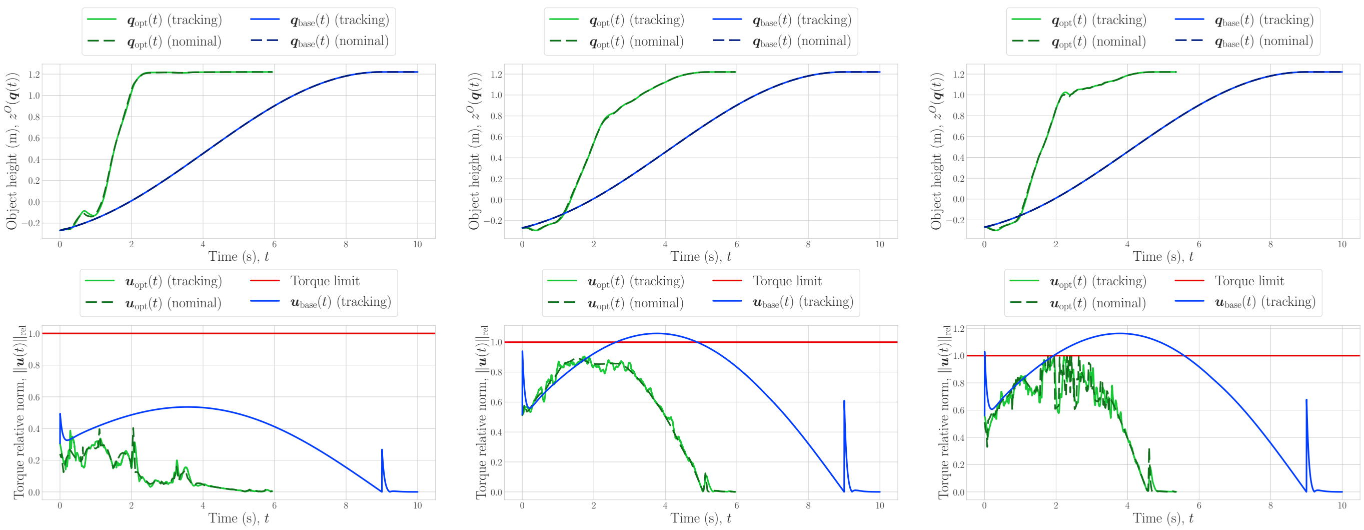
[1] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” 2019. [Online]. Available: https://drake.mit.edu

[2] C. R. Hargraves and S. W. Paris, “Direct trajectory optimization using nonlinear programming and collocation,” *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[3] P. E. Gill, W. Murray, and M. A. Saunders, “SNOPT: An SQP algorithm for large-scale constrained optimization,” *SIAM Rev.*, vol. 47, pp. 99–131, 2005.

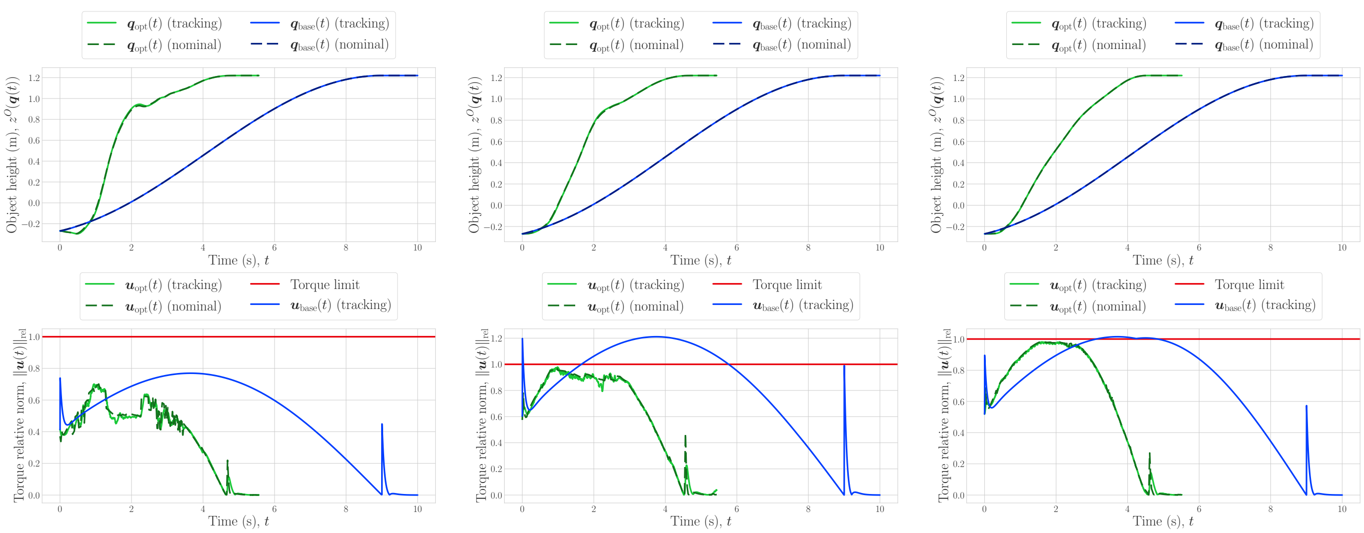
[4] P. E. Gill, W. Murray, M. A. Saunders, and E. Wong, “User’s guide for SNOPT 7.7: Software for large-scale nonlinear programming,” Department of Mathematics, University of California, San Diego, La Jolla, CA, Center for Computational Mathematics Report CCoM 18-1, 2018.

[5] “Kuka sensitive robotics lbr iiwa,” p. 30, 2017. [Online]. Available: https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311bfd744b4b0eab25ca883f6d3/kuka_lbr_iiwa_brochure_en.pdf



(a) 5 kg ball weight (b) 15 kg ball weight (c) 17 kg ball weight

Fig. 1: The results of execution of the baseline trajectory and the optimized trajectory for lifting of ball weights of different masses. Top panels show the object height $z^O(\boldsymbol{q}(t))$ as a function of time. Bottom panels show the relative norm of the torque vector $\|\boldsymbol{u}(t)\|_{\text{rel}} = \max_i \dfrac{|[\boldsymbol{u}(t)]_i|}{[\boldsymbol{u}_{\text{lim}}]_i}$ as a function of time.



(a) 10 kg balanced dumbbell (b) 19 kg balanced dumbbell (c) 15 kg imbalanced (13.5 kg + 1.5 kg) dumbbell

Fig. 2: The results of execution of the baseline trajectory and the optimized trajectory for lifting of dumbbells with different configurations. Top panels show the object height $z^O(\boldsymbol{q}(t))$ as a function of time. Bottom panels show the relative norm of the torque vector $\|\boldsymbol{u}(t)\|_{\text{rel}} = \max_i \dfrac{|[\boldsymbol{u}(t)]_i|}{[\boldsymbol{u}_{\text{lim}}]_i}$ as a function of time.