2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Learning Energy-Efficient Trajectory Planning for Robotic Manipulators using Bayesian Optimization

Philipp Holzmann[1], Maik Pfefferkorn[1,2], Jan Peters[3,4,5], and Rolf Findeisen[1]

Abstract— **Energy-optimal operation of robotic systems has gained high interest in both industry and science. We propose to fuse model predictive control and Bayesian optimization to plan minimum-energy trajectories for industrial robots that guarantee successful executions of the primary task. Particularly, parts of the predictive planner are learned using Bayesian optimization to account for the secondary, higher-level objective – here energy minimization. The effectiveness of the proposed approach is underlined in simulation, where a reduction in energy consumption is observed while maintaining a high quality of task executions.**

I. Introduction

Nowadays, robots are deployed in many different industries, e.g., as part of automated manufacturing systems [1]. There are many reasons for this such as their accuracy, repeatability and speed of (repetitive) task executions [2]. However, the increasing deployment of industrial robots has led to an increase in the electrical energy consumed by manufacturing processes. Rising energy costs as well as the desire to become energy-neutral have increased the demand for reducing the energy consumption [3]. Furthermore, industries have to adapt to fluctuations in energy allocation and supply to account for flexible energy prices or energy supply limitations [4]. Thus, achieving maximum energy efficiency while allowing for a flexible adjustment of energy usage, e.g., by changing the production speed, is of paramount interest [5].

There exists a wide variety of approaches aiming towards energy efficiency of robotic manufacturing systems. Firstly, one can aim for an energy-efficient design of the manufacturing process, e.g., avoiding oversized robots for the intended task or reducing idle time when the robot is not in use [6]. Secondly, one can focus on the software side, e.g., path optimization, planning an energy-optimal trajectory that realizes the path, or implementing energy-saving standby modes that are used whenever the robot is idle [6].

We focus on this second class of approaches, considering a given robot, a particular task and a specified path for successful task completion given by a predefined contour. It remains to compute an energy-efficient trajectory that realizes the path exploiting the available degrees of freedom. For instance, avoiding high velocities and accelerations reduces energy consumption. However, this leads to long execution times and thus, increased energy consumption due to the required compensation of gravitational and payload forces. Moreover, different robot poses may lead to the same position on the path. We propose to compute energy-optimal trajectories using a predictive planners based on path-following model predictive control formulations [7], [8]. The planner solves an optimal control problem (OCP), thus being capable of directly accounting for constraints as well as the system dynamics. The planner decides about the timing of the reference path, thereby controlling the velocities, accelerations and task execution times. One of the key challenges is to select a suitable cost function for the predictive planner that achieves a desired accuracy on the path (primary objective) and energy optimality (secondary objective). These are (partially) competing subobjectives that need to be carefully balanced using (weighting) parameters. We propose to exploit the degrees-of-freedom given by the cost function parametrization to identify the task realization that is optimal with respect to the secondary objective while satisfying the primary objective. Specifically, we propose to use Bayesian optimization [9], a machine-learning-based optimization method for black-box objective functions, to learn the cost function parametrization based on measurement and simulation data from repeated task executions.

Optimization-based planners are widely used in literature to generate optimal trajectories for robotic systems [10], [11], [12]. They have been successfully applied to reduce energy consumption of robots [13], [14], [15], [16], [17], [18]. Current approaches rely either on an approximation of energy consumption exploiting velocities and accelerations [13] or on exact models of energy consumption that are employed directly as cost function in the planner or controller [14]. The special case of accumulated kinetic energy minimization, which, however, does not reflect the total energy consumption, can be treated using operational space controllers [19] in combination with reference attractors. Our approach directly accounts for the consumed energy while not relying on the availability of a possibly complex energy consumption model. Rather, we exploit Bayesian optimization to capture the energy consumption model implicitly from measurement data or expensive simulations that are not suited for direct use in the planner. Using Bayesian optimization to learn the cost function (its parametrization) of the planner allows to adapt to changing and uncertain operating conditions. Following these ideas, Bayesian optimization has recently been used to tune model predictive controllers based on closed-loop information [20], [21].

[1]Control and Cyber-Physical Systems Laboratory (CCPS), TU Darmstadt, {philipp.holzmann, maik.pfefferkorn, rolf.findeisen}@iat.tu-darmstadt.de
[2]Laboratory for Systems Theory and Automatic Control, Otto-von-Guericke University Magdeburg
[3]Intelligent Autonomous Systems Laboratory, TU Darmstadt, jan.peters@tu-darmstadt.de
[4]German Research Center for AI (DFKI), Research Department SAIROL
[5]Hessian Centre for Artificial Intelligence

The main contribution of this work is the integration of path-following optimal control and Bayesian optimization for planning energy-optimal trajectories. The proposed approach provides a flexible framework to exploit degrees-of-freedom in the task definition. While outlined for minimizing energy consumption, the framework can be applied straightforwardly to other scenarios. Thus, the proposed approach combines the main advantages of optimal control and Bayesian optimization: it directly accounts for constraints and the system dynamics while effectively and efficiently exploiting data to adapt without requiring in-depth prior knowledge. The effectiveness of the approach is underlined in simulation.

In Section II, we introduce the considered problem. Thereafter, the model predictive planner is described in Section III. In Section IV, we provide an overview of Gaussian process regression, followed by a discussion of Bayesian optimization in Section V. We introduce our proposed approach in Section VI and underline its effectiveness using simulations before concluding in Section VII.

II. Problem Formulation

We consider a nonlinear system – the robot –

$$\dot{x}(t) = f\big(x(t), u(t)\big), \qquad x(t_0) = x_0 \tag{1a}$$

$$y(t) = h\big(x(t)\big), \tag{1b}$$

where $x \in \mathbb{R}^{n_\mathrm{x}}$ is the system state (the angular positions and velocities of the robot), $u \in \mathbb{R}^{n_\mathrm{u}}$ the control input (the joint torques), $y \in \mathbb{R}^{n_\mathrm{y}}$ the output (such as the coordinates of the tool center point), $f : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\mathrm{x}}$ captures the dynamics, and $h : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}^{n_\mathrm{y}}$ maps the state to the output. The initial state at time t_0 is given by $x_0 \in \mathbb{R}^{n_\mathrm{x}}$, and the states as well as the inputs are subject to constraints $(x, u) \in \mathcal{X} \times \mathcal{U}$ such as maximum velocities, or collision avoidance constraints. The system shall follow a path in the output-space (the robot's operational space) given by

$$\mathcal{C} = \{y = F(\theta) \in \mathbb{R}^{n_\mathrm{y}} \,|\, \theta \in [\theta_l, \theta_u] \subset \mathbb{R}, F : [\theta_l, \theta_u] \to \mathbb{R}^{n_\mathrm{y}}\} \tag{2}$$

with path parameter θ while reducing energy consumption.

To achieve this goal, we establish a planning algorithm based on model predictive control (MPC) that determines the minimum-energy state and input trajectories with respect to the system dynamics. At the same time, the planner is supposed to keep the path-following error to a (acceptable) minimum while guaranteeing constraint satisfaction. However, directly employing an energy-based cost functional in the MPC-based trajectory planner can result in a computationally expensive optimization problem or might be impossible if energy consumption models are unavailable. Thus, we consider a standard quadratic cost function in the MPC-based planner that takes deviations from the path, control energy and progress on the path into account. As this leads to competing control objectives, it remains to select the weight parameters for each objective. To do so, we employ Bayesian optimization that iteratively learns the weight parameters from data such that the energy consumed for executing the planned trajectory is minimized while accounting for the desired path-following accuracy.

III. Model Predictive Planner

We employ a model predictive trajectory planner based on solving an optimal control problem to plan trajectories for the system (1) whose execution is safe, feasible and leads to the successful completion of the desired task. The advantage of the optimal-control-based trajectory planner is that it directly takes the system dynamics as well as constraints into account. Furthermore, it works for a wide variety of cost functions that enable to encode different control objectives. We define the underlying optimal control problem as

$$\min_{\bar{u}, \bar{v}, T} \quad J_\beta\big(\bar{y} - F(\theta), \bar{x}, \bar{u}, \bar{\theta}, \bar{v}, T\big) \tag{3a}$$

$$\text{s.t.} \quad \forall \tau \in [0, T],$$

$$\dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(0) = x(0), \tag{3b}$$

$$\dot{\bar{\theta}}(\tau) = \rho(\bar{\theta}(\tau), \bar{v}(\tau)), \quad \bar{\theta}(0) = \theta(0), \tag{3c}$$

$$\bar{y}(\tau) = h(\bar{x}(\tau)), \tag{3d}$$

$$\bar{x}(\tau) \in \mathcal{X}, \ \bar{u}(\tau) \in \mathcal{U}, \tag{3e}$$

$$\bar{\theta}(\tau) \in [\theta_l, \theta_u], \ \bar{v}(\tau) \in \mathcal{V}, \tag{3f}$$

$$T \in [0, T_u], \tag{3g}$$

$$\big[\bar{x}^\top(T) \quad \bar{\theta}(T)\big]^\top \in \Omega. \tag{3h}$$

Therein, (3b) and (3d) represent the robot dynamics (1) and $\bar{\cdot}$ denotes a prediction inside the optimal control problem. The path parameter θ, defining the output reference according to (2), is equipped with a virtual dynamics $\rho : \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_\theta}$ (3c) controlled by the virtual input $\bar{v}(t)$ [8]. Since we optimize over both the system input $\bar{u}$ and the virtual input $\bar{v}$, the optimizer can control the temporal evolution of the path parameter, given by (3c), and hence the timing of the output reference to be tracked. The virtual dynamics are a design choice and often implemented as an integrator chain. We consider constraints on the system states and control inputs (3e) as well as on the path parameter and the virtual inputs (3f). We constrain the final time T within which the task has to be completed by (3g). Furthermore, a terminal constraint (3h) is deployed, which may be – together with the terminal cost (to be defined shortly) and a terminal controller – used to establish convergence guarantees [7].

We define the cost function (3a) as

$$J_\beta = \int_0^T L_\beta\big(\bar{y} - F(\theta), \bar{x}, \bar{u}, \theta, \bar{v}, T\big)\, d\tau + E\big(\bar{x}(T), \theta(T)\big), \tag{4}$$

where $E(\cdot)$ and $L_\beta(\cdot)$ are the terminal and stage cost, respectively. The particular structure of the cost function J_β depends on the task for which the trajectories are planned. Usually, the cost is a weighted sum of multiple, potentially competing subobjectives. The weighting parameters $\beta \in \mathbb{R}^{n_\beta}$ adjust the importance of the subobjectives in regard of the overall task objective and are often determined by human experts. However, there is usually no unique set of weighting parameters that leads to a successful task completion. Hence, there is a degree-of-freedom in tuning β, which we exploit to achieve a higher-level objective, e.g., minimizing the energy

consumption. We use Gaussian process (GP) regression to learn the energy consumption as a function of β, which is later optimized using Bayesian optimization.

IV. Gaussian Process Regression

A Gaussian process, denoted by

$$g(\xi) \sim \mathcal{GP}(m(\xi), k(\xi, \xi')) \tag{5}$$

defines a Gaussian probability distribution over functions. It is defined by its prior mean function $m : \mathbb{R}^{n_\xi} \to \mathbb{R}, \xi \mapsto \mathrm{E}[g(\xi)]$ and prior covariance function (kernel) $k : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_\xi} \to \mathbb{R}, (\xi, \xi') \mapsto \mathrm{Cov}[g(\xi), g(\xi')]$. Formally, it is defined as a collection of random variables – the function values $g(\xi)$ for all $\xi \in \mathbb{R}^{n_\xi}$ – any finite number of which is jointly and consistently Gaussian distributed [22].

We employ a Gaussian process g to model an unknown function $\varphi : \mathbb{R}^{n_\xi} \to \mathbb{R}, \xi \mapsto \varphi(\xi)$. To this end, we rely on a set $\mathcal{D} = \{(\xi_i, \gamma_i = \varphi(\xi_i) + \varepsilon) \,|\, i = 1, \dots, n_\mathcal{D}\}$ of $n_\mathcal{D}$ noisy observations $\gamma_i = \varphi(\xi_i) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma_\mathrm{n}^2)$ models white Gaussian measurement noise with variance σ_n^2. Therein, $(\gamma_1, \dots, \gamma_{n_\mathcal{D}})$ realizes the random vector $(g(\xi_1), \dots, g(\xi_{n_\mathcal{D}}))$. For conciseness of notation, we write in the following $\mathcal{D} = \{\Xi, \gamma\}$, where $\Xi \in \mathbb{R}^{n_\mathcal{D} \times n_\xi}$ is the training input matrix and $\gamma \in \mathbb{R}^{n_\mathcal{D}}$ is the training target (output) vector. Given that we want to predict a set of test targets $\varphi(\xi_i^*), i = 1, \dots, n_\mathcal{D}^*$ at test inputs ξ_i^*, concisely denoted as $\{\Xi^* \in \mathbb{R}^{n_\mathcal{D}^* \times n_\xi}, \varphi^* \in \mathbb{R}^{n_\mathcal{D}^*}\}$, we consider the prior distribution[1]

$$\begin{bmatrix} g(\Xi) \\ g(\Xi^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\Xi) \\ m(\Xi^*) \end{bmatrix}, \begin{bmatrix} k(\Xi, \Xi) & k(\Xi, \Xi^*) \\ k(\Xi^*, \Xi) & k(\Xi^*, \Xi^*) \end{bmatrix} \right). \tag{6}$$

We obtain the posterior (predictive) distribution for the test targets by conditioning (6) on $\mathcal{D}$, yielding $g(\Xi^*) \,|\, \Xi^*, \Xi, \gamma \sim \mathcal{N}(m^+(\Xi^*), k^+(\Xi^*, \Xi^*))$ with

$$m^+(\Xi^*) = m(\Xi^*) + k(\Xi^*, \Xi) k_\gamma^{-1} (\gamma - m(\Xi)) \tag{7a}$$

$$k^+(\Xi^*, \Xi^*) = k(\Xi^*, \Xi^*) - k(\Xi^*, \Xi) k_\gamma^{-1} k(\Xi, \Xi^*), \tag{7b}$$

where $k_\gamma = [k(\Xi, \Xi) + \sigma_\mathrm{n}^2 \mathbb{I}]$. The posterior mean (7a) is an estimate for the unknown test targets, i.e., $\hat{\varphi}^* = m^+(\Xi^*)$, while the posterior variances, i.e., the diagonal elements of (7b), quantify prediction uncertainty [22].

Usually, the prior mean and covariance functions $m(\cdot; \theta)$ and $k(\cdot, \cdot; \theta)$ depend on free hyperparameters θ. These hyperparameters have to be adjusted to the underlying problem in order to obtain meaningful predictive distributions. One possibility to find suitable hyperparameters is to infer them from the training data using evidence maximization [22]. In evidence maximization, we maximize the expressiveness of the GP model, given by the logarithmic marginal likelihood

$$\log\big(p(\gamma \,|\, \Xi, \theta)\big) = -\frac{1}{2} \gamma_0^\top k_\gamma^{-1} \gamma_0 - \frac{1}{2} \log(|k_\gamma|) - \frac{n_\mathcal{D}}{2} \log(2\pi), \tag{8}$$

where $\gamma_0 = \gamma - m(\Xi)$, $p(\cdot)$ denotes a probability density function and $|\cdot|$ the determinant. The optimal hyperparameters are then obtained from maximizing (8) w.r.t. θ.

[1] With a slight abuse of notation, we overload functions g, m and k in the following and mean by $g(A)$, $m(A)$ column vectors defined by $[g(A)]_i = g(a_i)$, $[m(A)]_i = m(a_i)$ and by $k(A, B)$ a matrix defined by $[k(A, B)]_{ij} = k(a_i, b_j)$. Therein, $A, B \in \{\Xi, \Xi^*\}$ and a_i, b_j denote the i^{th} and j^{th} row of A and B respectively.

V. Bayesian Optimization

Bayesian optimization (BO) is an optimization method for solving problems of the form

$$\xi^\dagger = \arg \max_{\xi \in \mathcal{R} \subseteq \mathbb{R}^{n_\xi}} \{\varphi(\xi)\}, \tag{9}$$

where we seek the global optimizer $\xi^\dagger$ that maximizes the cost function $\varphi : \mathbb{R}^{n_\xi} \to \mathbb{R}$ over a set $\mathcal{R}$. In Bayesian optimization, we assume an unknown cost function φ that, however, can be observed (evaluated) at query points $\xi_\mathrm{q} \in \mathcal{R}$. Evaluating φ may be (computationally) expensive, φ might be complex, e.g., nonlinear and nonconvex, or gradient information about φ may be lacking. This often prevents from deploying standard optimization approaches such as widely used gradient-based methods [9], [23].

To remedy from this situation, Bayesian optimization relies on a Bayesian surrogate model of the cost function, for which Gaussian processes are a common choice. Thus, we place a Gaussian process prior (5) over φ, representing our beliefs about the cost function we are optimizing. In Bayesian optimization, this surrogate model is now sequentially refined by iterating the following two steps:

(i) Select a query point ξ_n^q at which the cost function is probed to generate a new observation $\{\xi_n^\mathrm{q}, \varphi(\xi_n^\mathrm{q})\}$, and
(ii) Update the (posterior) GP model given the augmented training data set $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n \cup \{\xi_n^\mathrm{q}, \varphi(\xi_n^\mathrm{q})\}$.

Therein, subscript $n \in \mathbb{N}_0$ denotes the dependency on the current iteration and $\mathcal{D}_n$ is the set of training data collected up to iteration n [9], [23].

The crucial step in the iterative procedure described above is the selection of a query point ξ_n^q such that the sequential search is effectively guided towards the global optimizer $\xi^\dagger$, i.e., it is not necessary to probe any point in space but we want to achieve that $\xi_n^\mathrm{q} \xrightarrow{n \to \infty} \xi^\dagger$. To this end, acquisition functions are used. An acquisition function $\alpha : \mathbb{R}^{n_\xi} \to \mathbb{R}, \xi \mapsto \alpha(\xi; \mathcal{D}_n)$ utilizes the posterior GP model of φ to evaluate the utility of a query point, trading off exploration of the search space and exploitation of already identified, promising areas. The query point ξ_n^q is then determined, using the available information up to iteration n, as

$$\xi_n^\mathrm{q} = \arg \max_{\xi \in \mathcal{R}} \{\alpha(\xi; \mathcal{D}_n)\}. \tag{10}$$

While there exist many different acquisition functions [9], [23], we employ the so-called expected improvement acquisition function, given by

$$\begin{aligned} \alpha_\mathrm{EI}(\xi; \mathcal{D}_n) = (m_n^+(\xi) - \lambda + \omega) \bar{\Phi}\left(\frac{m_n^+(\xi) - \lambda}{\sqrt{k_n^+(\xi, \xi)}} \right) \\ + \sqrt{k_n^+(\xi, \xi)} \phi\left(\frac{m_n^+(\xi) - \lambda}{\sqrt{k_n^+(\xi, \xi)}} \right). \end{aligned} \tag{11}$$

Therein, m_n^+ and k_n^+ denote the posterior mean and covariance functions (7a) and (7b) based on $\mathcal{D}_n$. Furthermore, ϕ denotes the standard Gaussian probability density function and $\bar{\Phi}(\cdot) = 1 - \Phi(\cdot)$, where Φ denotes the standard Gaussian cumulative density function. The target value λ is the unknown, best reachable cost value. In practice, λ is often set to be the so-far best observed value, i.e., $\lambda = \lambda_n = \max\{\varphi(\xi_1^{\mathrm{q}}), \dots, \varphi(\xi_{n-1}^{\mathrm{q}})\}$. The parameter $\omega \geq 0$ may be used to enhance exploration.

The sequential approach of Bayesian optimization to solving Problem (9) using a series of query points, i.e., candidate optimizers, is in general data-efficient. However, note that obtaining the desired convergence guarantees has only been partially resolved in literature and is an ongoing area of research, see [9] and references therein. Nonetheless, Bayesian optimization has been deployed successfully in a wide variety of applications [24], [25].

VI. Energy-optimal Trajectory Planning for a Robotic Manipulator

We aim to find energy-optimal trajectories for a Franka Emika Panda robot performing a path-following task. Firstly, we discuss the modeling of the robot and the peculiarities of the trajectory planner. Afterwards, the BO-based algorithm that is used to learn the cost function of the planner is outlined. We conclude by presenting simulation results.

A. Trajectory Planner

An important aspect, when it comes to trajectory planning for robotic manipulators, are the dynamics of the system. Commonly, the dynamics are derived using Lagrangian mechanics in combination with a tailored approach for parameter estimation [26]. The resulting models are often given in implicit form as

$$\tau = B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) \tag{12}$$

where $\tau \in \mathbb{R}^n$ are the joint torques. The joint angles, velocities and accelerations are given by $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$, where n denotes the number of joints. The terms $B(q), C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ represent the contributions of the inertia and Coriolis forces to the torques, respectively, while $g(q) \in \mathbb{R}^n$ is the gravitational contribution to the torques. Due to the structure of (12), one can use feedback linearization to simplify the model used in the OCP (3b), c.f., [26]. Feedback linearization allows us to formulate our dynamics as a double integrator. Consequently, we define the states as $[x_1, x_2]^\top = [q, \dot{q}]^\top$ with input $u = \ddot{q}$. We constrain the state and input (3e) according to the joint limits of the manipulator. The output function (3d) is derived from the direct forward kinematics of the robot and given by

$$y = [h_{\mathrm{pos}}(x_1), h_{\mathrm{ori}}(x_1)]^\top. \tag{13}$$

The function $h_{\mathrm{pos}} : \mathbb{R}^n \to \mathbb{R}^3$ maps the joint angles to the Cartesian position of the end-effector, and $h_{\mathrm{ori}} : \mathbb{R}^n \to \mathbb{R}^4$ directly defines the end-effector orientation error based on quaternions. It is also possible to include additional output constraints in the planner, e.g., to maintain a fixed orientation.

We define the stage cost in (4) as

$$\begin{aligned} L_\beta =& \|(h_{\mathrm{pos}}(x_1) - F(\theta))\|_{\beta_1}^2 + \|h_{\mathrm{ori}}(x_1)\|_{\beta_2}^2 \\ &+ \|\theta\|_{\beta_3}^2 + \|v\|_{\beta_4}^2 + \|x_2\|_{\beta_5}^2 + \|u\|_{\beta_6}^2 + \|T\|_{\beta_7}^2, \end{aligned} \tag{14}$$

where we penalize the quadratic error between the end-effector position and the given path, and the orientation error. Furthermore, we penalize the progress of the path parameter θ, the artificial input v, the joint velocities x_2, the joint accelerations u, and the final time T of task completion. We set $E(x(T), \theta(T)) = 0$ as the terminal cost function (4). To guarantee that the planner predicts trajectories to the end of the path, we use as the terminal constraint in (3h) $[h_{\mathrm{pos}}(x_1(T)), \theta(T)]^\top = [F(\theta_u), \theta_u]^\top$ with $\theta \in [\theta_l, \theta_u]$.

The cost (14) is characterized by a set of weighting parameters. For this set of parameters $\beta = \{\beta_i \in \mathbb{R}_{>0} \mid i \in \mathcal{I}, \mathcal{I} = \mathcal{I}_f \cup \mathcal{I}_t\}$, we distinguish between two index sets. The index set $\mathcal{I}_f$ marks all the fixed parameters $\beta_{\mathcal{I}_f}$, while $\mathcal{I}_t$ represents the index set of the parameters $\beta_{\mathcal{I}_t}$ that we want to learn. Hence, the planner has additional degrees of freedom that we can use to impose superordinate objectives. For simplicity, we only consider three tuning parameters: β_4, β_6, and β_7. It is possible to exploit richer cost function parametrizations, such as using individual penalty parameters for joint velocities and accelerations, at the cost of higher computational loads as BO is known to suffer from the curse of dimensionality. We aim to adjust β such that the computed state and input trajectories result in a reduction of consumed energy during task execution.

B. Parameter Learning using Bayesian Optimization

We deploy Bayesian optimization to learn the free parameters of the primary cost function (14) of the planner. Specifically, we seek the optimal parameters $\beta^\dagger$ such that the trajectories resulting from solving OCP (3) with cost function (14) are energy-optimal. To this end, we quantify the energy consumption in task execution (iteration) j and given a set of parameters β^j as

$$W^j = W(\beta^j) = \int_0^T |{\tau^j}^\top(t)\dot{q}^j(t)|\mathrm{d}t, \tag{15}$$

where $\tau^j(t)$ and $\dot{q}^j(t)$ describe the trajectories of the joint torques and velocities, respectively, for $t \in [0, T]$.[2]

To model the function W in (15), we employ a zero-mean Gaussian process with squared-exponential covariance function. To this end, we rely on the training data set $\mathcal{D}_j$, which is initialized as $\mathcal{D}_1 = \emptyset$ and afterwards updated by $\mathcal{D}_j = \{(\beta^i, W^i) \mid i = 1, \dots, j-1\} = \mathcal{D}_{j-1} \cup \{(\beta^{j-1}, W^{j-1})\}$ for $j \geq 2$. Based thereon, the next candidate optimizer β^{j+1} to be probed in the next iteration is computed according to (10) using the expected improvement acquisition function (11). We summarize the algorithm in the following:

(i) Initialize $\mathcal{D}_1 = \emptyset$ and $\beta^1 = \beta^{\mathrm{init}}$.

[2]Exact energy models are often not available in practice. Note that (15) is an approximation since it does not account for the amount of energy required to hold a certain pose.

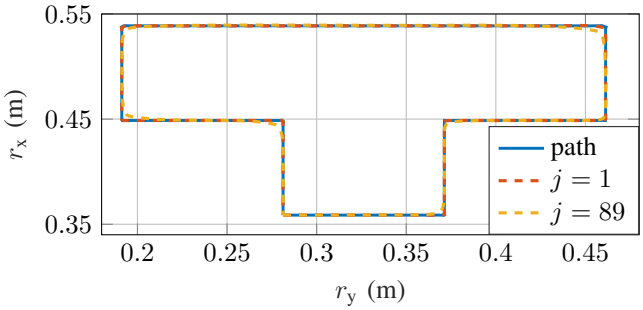

Fig. 1. Desired path (blue line) and path-following results for the first (red line) and final, energy-optimal (yellow line) run are shown.

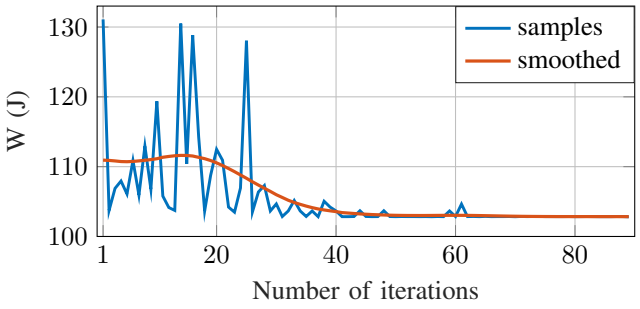

Fig. 2. Energy consumption over the BO iterations: Energy consumption values for each query parametrization (blue line) and smoothed version thereof (red line) are shown.

(ii) Solve OCP (3) using β^j to obtain task-optimal state and input trajectories $\mathbf{x}^\dagger$ and $\mathbf{u}^\dagger$, respectively.
(iii) Evaluate the energy consumption W^j (15) and augment the training data set by $\mathcal{D}_{j+1} = \mathcal{D}_j \cup \{(\beta^j_{\mathcal{I}_t}, W^j)\}$.
(iv) Update posterior GP model of W based on $\mathcal{D}_{j+1}$, including hyperparameter optimization.
(v) Exploit the updated GP model to obtain the next candidate optimizer $\beta^{j+1} = \beta^{j+1}_{\mathcal{I}_t} \cup \beta^{j+1}_{\mathcal{I}_f}$, where $\beta^{j+1}_{\mathcal{I}_t} = \arg\max_\beta \{\alpha_{\text{EI}}(\beta; \mathcal{D}_j)\}$ according to (10) with acquisition function (11).
(vi) If $j \geq N_{\min}$ and $|W_j - W_{j-1}| \leq \rho$ for a small constant $\rho \in \mathbb{R}_+$ and a minimum number $N_{\min}$ of iterations, terminate and set $\beta^\dagger = \beta^j$. Otherwise, go to (ii) with $j \leftarrow j + 1$.

In the following, we demonstrate the effectiveness of the algorithm in simulation. The algorithm has been implemented in Matlab using GPML [22] and CasADi [27].

C. Minimum-energy Trajectories for a Robotic Manipulator

We consider a seven-link Franka Emika Panda robot[3]. We exploit the dynamic model derived in [28]. The robot is supposed to follow a predefined path in the Cartesian (r_x, r_y)-plane with its end-effector, see Fig. 1. We apply the approach outlined in the previous section to compute energy-optimal trajectories for the robot by learning $\beta_{\mathcal{I}_t}$. The energy decrease over the iterations is shown in Fig. 2.

Within the first 30 iterations, we observe that the energy cost evaluation is very noisy with high peaks. This is due to the exploration and exploitation trade-off. In the beginning, the BO tries to explore the parameter space more to gain information about the underlying cost function. From iteration 31 and forward, the peaks become smaller because the BO relies more on exploitation from this point on. At the 89th iteration step, the algorithm converges to the minimal value of (15) and we found the optimal parameters β^*. From Fig. 2, one can deduce how challenging it can be for an expert operator to find an optimal parameter set. It would require many trial and error experiments. Bayesian optimization, however, can find a global optimum and does not require extensive hands-on tuning. Furthermore, the proposed approach can be extended to situations where knowledge about the system or its environment is limited, using a fully parameterized model predictive planner, cf. [9].

Given the optimal parameters, the trajectory planner (3) determines the energy-optimal state and input trajectories for following the predefined path with acceptable accuracy. The path is the contour of a polygon that lies in the r_x-r_y-plane, where the r_z-coordinate is constant. Fig. 1 shows the path, the actual end-effector position in the first iteration, and the actual end-effector position in the 89th iteration using the energy-optimal parameters in the planner. We see that the robot is able to follow the path with small errors in the first iteration. When using the energy-optimal parameters, we observe that the end-effector is navigated more smoothly in the corners of the given contour. This is due to the fact that we used the fixed parameters $\beta_{\mathcal{I}_f}$ to prioritize low position errors in the planner. A more detailed comparison of the errors and energy consumption is given in Tab. I. Here, the error is defined as the Euclidean distance between the path and the end-effector position. The maximum errors and average errors are given by $\|e\|_{\max}$ and $\|e\|_{\text{avg}}$, respectively.

The corresponding joint velocity, acceleration, and torque trajectories are shown in Fig. 3 for the second joint. Here, we observe that the absolute value of the torques becomes smaller over the iterations and we have smoothing effects on the torques, especially when comparing to the result of the 15th iteration, which displays large spikes in the torques. Such abruptly changing torques impose a strain on the joint drives, which leads to promoted fatigue effects and decreased longevity. Hence, besides not being energy-optimal, such trajectories are to be avoided in order to reduce wear and tear. Furthermore, we see that the trajectories also vary in the task execution time since the weighting parameter for penalizing larger task execution times in (14) is adapted by the BO procedure. The tuning algorithm tries to find a trade-off between the joint velocities and accelerations and the maximum task execution time that ultimately leads to a decrease of the overall energy. Also important to note is that the trajectory in iteration 15 ends at exactly 25s. This is due the hard constraint on the final time in (3g).

TABLE I
PATH-FOLLOWING ERROR AND ENERGY CONSUMPTION.

No. iteration	$\|e\|_{\max}$ (mm)	$\|e\|_{\text{avg}}$ (mm)	W (J)
1	1.5	0.03	131
89	5	0.6	103

[3]The joint constraints can be found at: `frankaemika.github.io/docs/control_parameters.html`

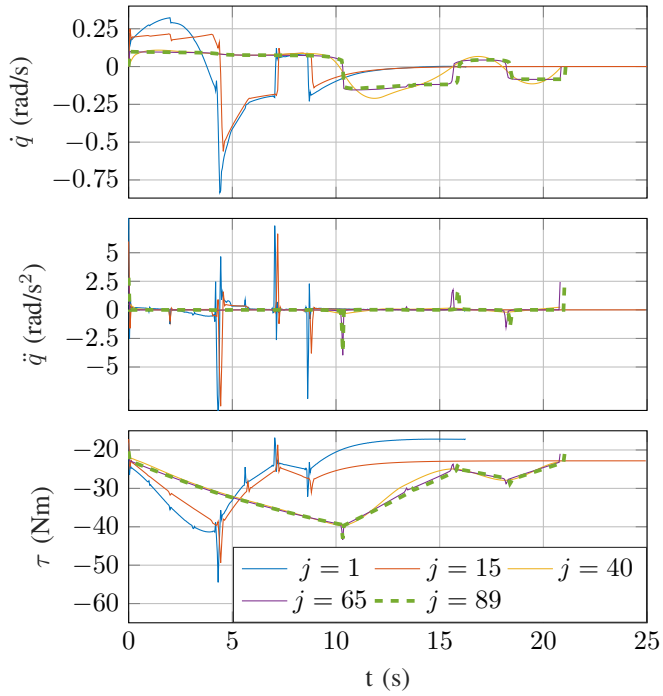

Fig. 3. Exemplary results of the energy-optimal joint velocity (top), joint acceleration (middle), and joint torque (bottom) trajectories of the second joint for indicated BO iterations.

VII. Conclusions

We presented a trajectory planning approach that solves a path-following OCP to compute a realization of a specified task. Besides this primary objective, encoded directly in the OCP, we account for secondary, higher-level objectives by exploiting degrees-of-freedom in the form of learning parameters of the OCP using Bayesian optimization. The proposed approach was used to compute minimum-energy trajectories for a robotic manipulator for a path-following task. We have demonstrated a significant reduction in energy consumption, while the planned trajectories also led to a successful completion of the primary task, illustrating the effectiveness of the proposed approach.

Future research will be dedicated towards implementing the proposed approach for a real-world scenario. Furthermore, different higher-level objectives besides minimizing energy consumption can be considered, such as quality of task executions (e.g., quality of a welding seam). Future steps include further the extension of the proposed scheme to scenarios subject to uncertainty and to safety-critical scenarios in which constraints need to be satisfied in each iteration.

References

[1] R. Goel and P. Gupta, "Robotics and industry 4.0," in *A Roadmap to Industry 4.0: Smart Production, Sharp Busines and Sustainable Development*. Springer, 2020, pp. 157–169.

[2] T. Chettibi, H. E. Lehtihet, M. Haddad, and S. Hanchi, "Minimum cost trajectory planning for industrial robots," *European Journal of Mechanics - A/Solids*, vol. 23, no. 4, pp. 703–715, 2004.

[3] M. Schule, H. Nehler, M. Ottosson, and P. Thollander, "Energy management in industry – a systematic review of previous findings and an integrative conceptual framework," *J. Clean Prod.*, vol. 112, pp. 3692–3708, 2016.

[4] H. Reinhardt, J.-P. Bergmann, M. Münnich, D. Rhein, and M. Putz, "A survey on modeling and forecasting the energy consumption in discrete manufacturing," *Procedia CIRP*, vol. 90, pp. 443–448, 2020.

[5] P. Paryanto, M. Brossog, M. Bornschlegl, and J. Franke, "Reducing the energy consumption of industrial robots in manufacturing systems," *Int. J. Adv. Manuf. Tech.*, vol. 78, pp. 1315–1328, 2015.

[6] M. Soori, B. Arezoo, and R. Dastres, "Optimization of energy consumption in industrial robots, a review," *Cog. Rob.*, vol. 3, pp. 142–157, 2023.

[7] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *Proc. IEEE Conf. Dec. Cont.*, 2009, pp. 8642–8647.

[8] J. Matschek, T. Bäthge, T. Faulwasser, and R. Findeisen, "Nonlinear predictive control for trajectory tracking and path following: An introduction and perspektive," in *Handbook of Model Predictive Control*, S. Rakovic and S. Levine, Eds. Birkhäuser, 2019, pp. 169–198.

[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[10] M. Ghasemi, N. Kashiri, and M. Dardel, "Path planning for autonomous vehicles using model predictive control," *Proc. Inst. Mech. Eng. Part C: J. Mech. Eng. Scie.*, vol. 226, pp. 473–484, 2012.

[11] M. Massaro, S. Lovato, M. Bottin, and G. Rosati, "An optimal control approach to the minimum-time trajectory planning of robotic manipulators," *Robotics*, vol. 12, p. 64, 2023.

[12] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *Int. Conf. Rob. Auto.*, 2011, pp. 3719–3726.

[13] S. Björkenstam, D. Gleeson, R. Bohlin, J. S. Carlson, and B. Lennartson, "Energy Efficient and Collision Free Motion of Industrial Robots using Optimal Control," in *Conf. Aut. Sci. Eng.*, 2013, pp. 510–515.

[14] C. Garriz and R. Domingo, "Trajectory Optimization in Terms of Energy and Performance of an Industrial Robot in the Manufacturing Industry," *Sensors*, vol. 22, no. 19, p. 7538, 2022.

[15] K. Paes, W. Dewulf, K. Vander Elst, K. Kellens, and P. Slaets, "Energy efficient trajectories for an industrial ABB robot," *Procedia Cirp*, vol. 15, pp. 105–110, 2014.

[16] F. Stuhlenmiller, D. Clever, S. Rinderknecht, M. Lutter, and J. Peters, "Trajectory optimization of energy consumption and expected service life of robotic systems," in *Int. Conf. Adv. Int. Mechat.*, 2021, pp. 842–847.

[17] H. Potter, J. Kem, G. Gonzalez, and C. Urrea, "Energetically optimal trajectory for a redundant planar robot by means of a nested loop algorithm," *Elektronika ir Elektrot.*, vol. 28, no. 2, pp. 4–17, 2022.

[18] O. von Stryk and M. Schlemmer, "Optimal Control of the Industrial Robot Manutec r3," in *Computational Optimal Control*, R. Bulirsch and D. Kraft, Eds. Birkhäuser, 1994, pp. 367–382.

[19] K. Kazerounian and Z. Wang, "Global versus local optimization in redundancy resolution of robotic manipulators," *Int. J. Robot. Res.*, vol. 5, no. 2, pp. 3–12, 1988.

[20] Q. Lu, R. Kumar, and V. M. Zavala, "MPC Controller Tuning using Bayesian Optimization Techniques," *arXiv:2009.14175*, 2020.

[21] F. Sorourifar, G. Makrygirgos, A. Mesbah, and J. A. Paulson, "A Data-Driven Automatic Tuning Method for MPC under Uncertainty using Constrained Bayesian Optimization," *IFAC-Papers Online*, vol. 54, no. 3, pp. 243–250, 2021.

[22] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: MIT Press, 2006.

[23] P. I. Frazier, "A Tutorial on Bayesian Optimization," *arXiv:1807.02811*, 2018.

[24] D. Ulmasov, C. Baroukh, B. Chachuat, M. P. Deisenroth, and R. Misener, "Bayesian optimization with dimension scheduling: Application to biological systems," in *26th Europ. Symp. Comp. Aided Chem. Eng.*, vol. 38, 2012, pp. 1051–1056.

[25] Y. Jin and P. V. Kumar, "Bayesian optimisation for efficient material discovery: a mini review," *Nanoscale*, vol. 15, pp. 10 975–10 984, 2023.

[26] B. Siciliano, S. Lorenzo, V. Luigi, and O. Giuseppe, *Robotics: Modeling, Planning and Control*. Springer, 2010.

[27] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Math. Prog. Comp.*, vol. 11, no. 1, pp. 1–36, 2019.

[28] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordono, and A. D. Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Rob. Aut. Let.*, vol. 4, no. 19, pp. 4147–4154, 2019.