Distributed Co-Design of Motors and Motions for Robotic Manipulators

Zehui Lu, Yebin Wang, Yusuke Sakamoto, and Shaoshuai Mou

Abstract—This paper studies a manipulator co-design problem of motors and motions for multiple tasks. To reduce computational burden and improve scalability as the number of tasks grows, this paper introduces a distributed co-design framework to handle the co-design process for all tasks in a distributed fashion. Moreover, this paper presents a distributed constrained optimization algorithm, which secures a unified set of design parameters for all tasks ultimately such that the total average of motor operation efficiency is optimized and the design constraints are satisfied across all tasks and motors. The distributed manner reduces the computational load by allowing each agent to solve co-design optimization solely for its designated task. A numerical simulation further verifies the proposed algorithm.

I. INTRODUCTION

Off-the-shelf robotic manipulators typically have specified features to meet diverse user needs. These robots are typically designed to cater to a wide range of tasks, which implies optimality or sub-optimality for a particular performance index. Such an application-oriented and optimization-based robot design can lead to efficient solutions in terms of energy usage, cost-effectiveness, and productivity. The robot design is naturally multidisciplinary, involving structural and geometric design [1]–[3], battery sizing [4], kinematics [5], dynamics [2], [6], and control [7], [8]. The design objectives are also multidisciplinary, including weight [1], [9], energy consumption [10], task completion time [10], [11], etc.

The process of co-design, which addresses the interdependence and conflicts between subsystems during the design phase [12], has the potential to mitigate sub-optimal outcomes that may arise from a design process focused on individual disciplines. The concept of co-design is prevalent in numerous applications, such as general robotic module selection [12], robotic manipulators [1], [7], [13], legged robots [14]–[16], medical robots [3], soft robots [17], etc. In particular, [1]–[3] optimizes the structure, inertia, and shape of a robot. [7] co-designs the drivetrain and joint trajectories of a manipulator, where the drivetrain is parameterized by motor shaft length and gearbox ratio. [8] performs co-design of mechanical plant parameters and optimal feedback control strategies. [9] optimizes the choice of gearbox, motor drive train, and parameters for the manipulator’s links. [10], [18] perform simultaneous optimization for both trajectory and controller design. [19] co-optimizes for motions, robot physical parameters, and motor selection. [20] presents a bi-level optimization framework to determine the optimal hardware parameters for a legged robot and an energy-efficient trajectory for a given task, operating at two distinct levels. Moreover, there has been a growing interest in task-specific robot co-design derived from high-level user specifications. Existing work includes co-design of motion and physical parameters for a single task [1], [2], [7]–[9], [19] and for multiple tasks [10], [18].

A primary obstacle in robot co-design is the significant computational load [14]. In practice, robot co-design needs to consider the specifications of multiple customers at once, due to the nature of manufacturing. Thus, co-design becomes more complicated when multiple objectives must be considered. To tackle this issue, [10], [18] propose a centralized bi-level stochastic programming framework for the co-design under multiple tasks. This framework considers task versatility by incorporating an overall expected cost function that accounts for the probability distribution of task occurrence rates. Specifically, at the outer level, the robot design is optimized by minimizing the average cost across all tasks. At the inner level, the robot’s motion is optimized independently for each task. Such a centralized framework aggregates information from all tasks and can pose computational challenges as the number of tasks grows, no matter whether the design parameters and the motion are determined simultaneously [3] or sequentially [10], [18]. Different from the centralized bi-level framework above, [21] proposes a consensus-based distributed bi-level optimization framework, which cooperatively adjusts the motion from each agent’s trajectory planner to further optimize an additional performance index.

Motivated by the distributed bi-level framework mentioned earlier, this paper aims to co-design the motions (joint position and velocity trajectories) and motor design parameters for a robotic manipulator with \( n \) degree-of-freedom (DOF) to handle multiple tasks in a distributed fashion. To achieve this, each task is conceptualized as an agent within a network. Given a task and arbitrary motor design parameters, a trajectory of joints’ angular positions and velocities can be computed by a local trajectory planner that aligns with the task requirements. Subsequently, a local loss function is used to measure the efficiency of each agent’s motors based on their respective trajectory. Due to manufacturing limitations, a unified set of design parameters is ultimately required to
fulfill all tasks, thereby optimizing the global average of each task’s local loss. This requirement of common design parameters can be posed as a consensus across the network.

Therefore, as illustrated in Fig. 1, this paper introduces a distributed co-design framework and proposes a distributed optimization algorithm based on the combination of the alternating direction method of multipliers (ADMM) and the augmented Lagrangian method (ALM). The aim is to compute optimal and unified design parameters, optimizing the global average of local loss functions in a distributed and iterative manner to effectively fulfill multiple tasks. The distributed manner reduces the computational load by allowing each agent to solve co-design optimization solely for its designated task. This framework ensures the scalability of the multi-task co-design as the number of tasks grows. The contributions of this paper are summarized as follows:

1) A distributed framework for manipulator co-design of motors and motions with multiple tasks;
2) A distributed constrained optimization algorithm to solve the distributed co-design problem.

Notations. The non-negative integer set is denoted by $\mathbb{Z}_+$. For $x, y \in \mathbb{R}^n$, $x \leq y$ indicates element-wise inequality. Let $\text{col}\{v_1, \cdots, v_n\}$ denote a column stack of elements $v_1, \cdots, v_n$, which may be scalars, vectors or matrices, i.e., $\text{col}\{v_1, \cdots, v_n\} = [v_1^\top \cdots v_n^\top]^\top$. Let $\otimes$ denote the Kronecker product. Let $0_n, 1_n \in \mathbb{R}^n$ denote a zero and an one vector. Let $I_n \in \mathbb{R}^{n \times n}$ denote an identity matrix.

II. System Modeling

This section introduces the modeling of an arbitrary surface permanent magnet synchronous motor’s (SPMSM) dynamics and an arbitrary $n$-DOF open-chain manipulator dynamics with $n$ SPMSMs.

A. SPMSM Modeling and Design Parametrization

This subsection presents the dynamical modeling of SPMSM with several motor design variables. The motor design variables are summarized in Table I. Fig. 2 further illustrates the physical meaning of the design variables, where the axial length $l$ is not shown. Denote an arbitrary motor’s design variables as

$$\beta \triangleq \text{col}\{l, r_{ro}, r_{so}, h_m, h_{sy}, w_{tooth}, b_0\} \in \mathbb{R}^7.$$ 

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$ Axial length of core</td>
<td>$h_m$ Height of magnet</td>
</tr>
<tr>
<td>$r_{ro}$ Outer radius of rotor</td>
<td>$w_{tooth}$ Width of tooth</td>
</tr>
<tr>
<td>$r_{so}$ Outer radius of stator</td>
<td>$b_0$ Stator yoke</td>
</tr>
<tr>
<td>$h_{sy}$ Width of stator yoke</td>
<td>Slot opening</td>
</tr>
</tbody>
</table>

Units of all design parameters are mm

Given the motor design parameters for one motor, the magnetic equivalent circuit (MEC) modeling technique is used to compute the necessary parameters for the motor dynamics analytically. The derivation details are summarized in Appendix A. The dynamic model of an arbitrary SPMSM can be written as follows [22]:

$$\frac{di_d}{dt} = -\frac{R}{L} i_d + \frac{p}{2} w \omega i_q + \frac{u_d}{L}, \quad (1a)$$

$$\frac{di_q}{dt} = -\frac{R}{L} i_q - \frac{p}{2} \omega (i_d + \frac{\Phi_m}{L}) + \frac{u_q}{L}, \quad (1b)$$

where $i_d$ and $i_q$ are the currents in the d- and q-axis, respectively; $u_d$ and $u_q$ are the voltages in the d- and q-axis, respectively; $\omega$ is the motor’s angular velocity. $p$ is referred to Appendix A. $R, L_d, L_q, \Phi_m$ are calculated by (22), (23) and (26) in Appendix A with $\beta$. The motor design variables are subject to the following constraints:

- $l \in [20, 100]$, $r_{ro} \in [10, 100]$, $r_{so} \in [10, 100]$, $h_m \in [1, 5]$, $h_{sy} \in [5, 10]$, $w_{tooth} \in [5, 20]$, $b_0 \in [1, 10]$, $h_{sy} > 0$ mm, $D_{wire} \geq 0.6$ mm, $k_C > 0$, $0 < \arcsin\left(\frac{w_{tooth}}{2(r_{ro} + \delta)}\right) + \arcsin\left(\frac{b_0}{2(r_{ro} + \delta)}\right) \leq \frac{\pi}{4}$, $0$ kg $< m_{stator} + m_{rotor} \leq 3$ kg, $m_{stator} > 0$ kg, $0$ T $< \frac{k_F \Phi_1}{w_{tooth}} \leq 1.5$ T, $0$ T $< \frac{k_F \Phi_1}{\sqrt{h_{sy}}} \leq 1.5$ T,

where (2d) describe the minimal slot height, motor’s minimal wire diameter, and minimal Carter’s coefficient; (2e) and (2f) describe the tooth width bound and the motor weight bound, respectively; (2g) describes the magnetic flux bounds in the tooth and the stator yoke. $\delta$ and $h_{ss,j}$, $D_{wire,j}$, $m_{stator,j}$, $m_{rotor,j}$, $\Phi_1$, $k_F$, $k_C$ can be obtained and calculated by (16) - (21) and (24) - (27) of Appendix A. The motor is additionally subject to some operational constraints for all $t$:

- $-3$ A $\leq i_d \leq 0$ A, $-3$ A $\leq i_q \leq 3$ A, $(-3a)$
- $100$ V $\leq u_d \leq 100$ V, $-100$ V $\leq u_q \leq 100$ V. $(-3b)$
B. n-DOF Open-Chain Manipulator Dynamics

Assume that the flexibility and the backlashes of the gears can be ignored, and the \((j+1)\)-th motor of a manipulator is rigidly attached to the \(j\)-th link, i.e., its stator is rigidly attached to the \(j\)-th link and its rotor is coupled with the \(j+1\)-th link. Following the standard modeling approach in [23, Chapter 8], a general dynamic model for an arbitrary \(n\)-DOF open-chain manipulator can be written as

\[
M(\theta, \dot{\theta}) \ddot{\theta} + C(\theta, \dot{\theta}, \beta) \dot{\theta} + G(\theta, \beta) = \tau, \tag{4}
\]

where \(\theta, \dot{\theta}, \ddot{\theta} \in \mathbb{R}^n\) are the angular positions, velocities, and accelerations \([\text{rad/s}^2]\) of \(n\) joints, respectively; \(M(\theta, \beta), C(\theta, \dot{\theta}, \beta), G(\theta, \beta) \in \mathbb{R}^{n \times n}\) and \(\tau(\theta, \beta) \in \mathbb{R}^n\) are the inertia-related matrix, Coriolis forces, and gravitational force with the parameterization of SPMSM under \(\beta\), respectively. Denote \(\bar{H} \equiv H \otimes I_7\) for the \(n\)-DOF manipulator dynamics (4) with \(\bar{H} = H \otimes I_7\).

The articulated-body algorithm (ABA) [24, Chapter 7.3] is used to compute the manipulator’s forward dynamics with zero tip force and a constant gravitational acceleration \(g = 9.81 \text{ m/s}^2\). Details are summarized in [25, Algorithm 1]. The constraints on \(\theta, \dot{\theta}\) are given by:

\[
\theta \leq \theta \leq \bar{\theta}, \quad \dot{\theta} \leq \bar{\theta}, \tag{5}
\]

where \(\theta, \bar{\theta}, \dot{\theta} \in \mathbb{R}^n\) denotes the position lower and upper bound, velocity lower and upper bound, respectively.

Remark 1. Each task may require a different payload on the end-effector, which can be treated as a rigid body attached to the final link. Given the mass, center of mass, and gravitational inertia of a payload, one can compute its inertia-related matrix and then add it to the spatial inertia matrix of link \(n\). The manipulator’s forward dynamics can still be computed by ABA [24, Chapter 7.3].

C. Complete Manipulator Dynamics

Denote \(i_{d,j}\) and \(u_{d,j}\) as the current and voltage in the \(d\)-axis for the \(j\)-th motor of an arbitrary manipulator. Denote the state of all the motors by \(x_m \triangleq \text{col}\{i_{d,1}, \cdots, i_{d,n}, i_{q,1}, \cdots, i_{q,n}\} \in \mathbb{R}^{2n}\); similarly denote the control of all motors by \(u_m \triangleq \text{col}\{u_{d,1}, \cdots, u_{d,n}, u_{q,1}, \cdots, u_{q,n}\} \in \mathbb{R}^{2n}\). Combining the \(n\)-DOF manipulator dynamics (4) with \(n\) motor’s dynamics (1) and the connection among motor states, controls, velocities, and torques (28)-(32) of Appendix B, the complete dynamics for an arbitrary manipulator are written as

\[
\dot{x} = f(x(t), u(t), \beta), \tag{6}
\]

where \(x \triangleq \text{col}\{\theta, \dot{\theta}, x_m\} \in \mathbb{R}^{4n}\) and \(u \triangleq u_m \in \mathbb{R}^{2n}\).

D. SPMSM Design Optimization Formulation

This subsection formulates the SPMSM design as an optimization problem, which aims to optimize operation efficiency given a specific task and the respective motor trajectory. Given the task and the complete dynamics (6), a trajectory planner, which will be defined in (9), can generate a trajectory of optimal open-loop states \(x^*(t)\) and controls \(u^*(t)\). Then the desired trajectory of joint angular positions \(\theta_{\text{des}}(t)\) and velocities \(\dot{\theta}_{\text{des}}(t)\) for all motors can be directly obtained from \(x^*(t)\) and \(u^*(t)\). From the description below (32) of Appendix B, a trajectory of each motor’s velocity \(\omega(t)\) and motor torque \(\tau_m(t)\) can be directly obtained. Consequently, each motor’s maximum (magnitude) velocity \(\omega_{\text{max}}\) and torque \(\tau_{\text{max}}\) are determined.

Then for each motor, given the operational data \(\xi \equiv \{x^*(t), u^*(t), \forall t\}\), one can measure the probability of each grid \((\tau_m, \omega)\) within the range \(A \triangleq [0, \tau_{\text{max}}] \times [0, \omega_{\text{max}}]\) by a two-dimensional probability density function (PDF) \(q(\tau_m, \omega; \beta)\). The details to generate a PDF are summarized in Algorithm 1, where \(N_\tau, N_\omega \in \mathbb{Z}_+\) are the grid numbers for the discretization of motor torque and velocity, respectively. Line 9 of Algorithm 1 converts a single operational data point \((x^*(t), u^*(t))\) at a time instant \(t\) to a grid \((\tau_m, \omega)\), i.e., (28)-(32) of Appendix B.

Next, the efficiency \(\eta(\tau_m, \omega; \beta)\) at each grid \((\tau_m, \omega)\) can be calculated by (33), Appendix B. Thus the design optimization for one motor is written as:

\[
\begin{align*}
\min_{\beta} & \quad \int_A q(\tau_m, \omega; \beta)(1 - \eta(\tau_m, \omega; \beta)) d\omega d\tau_m \\
\text{s.t.} & \quad h_{\Omega}(\beta) \leq 0, \quad h_{\Omega}(\beta, \tau_m, \omega) \leq 0, \quad \forall (\tau_m, \omega) \in A,
\end{align*} \tag{7}
\]

where \(\int_A\) denotes the double integration \(\int_0^{\tau_{\text{max}}} \int_0^{\omega_{\text{max}}}\) among \(A \triangleq [0, \tau_{\text{max}}] \times [0, \omega_{\text{max}}]\); \(\tau_{\text{max}}\) and \(\omega_{\text{max}}\) are determined by \(\xi\); the motor design constraints \(h_{\Omega}(\beta)\) are summarized as (2); the motor operational constraints \(h_{\Omega}(\xi, \beta)\) are summarized as (3), where \(i_{d}, i_{q}, u_{d}, u_{q}\) are determined by (35) given each \((\tau_m, \omega)\) within \(A\). Note that evaluating the efficiency of a motor operation PDF is more robust and numerically stable than of a specific trajectory.

III. DISTRIBUTED CO-DESIGN FORMULATION

This section introduces the formulation of a distributed consensus-based co-design problem. Consider a network of \(N_a\) agents labeled as \(V = \{1, \cdots, N_a\}\), where each agent \(i\) determines the design of manipulator \(i\) based on a given task \(i\). Agent \(i\) can receive information from its neighbor set \(N_i\). \(G = \{V, E\}\) denotes an undirected graph such that an undirected edge \((i, r) \in E\) if and only if \(i\) and \(r\) are neighbors. \(E \subseteq \mathbb{R}^{N_x \times N_u}\) denotes the Laplacian matrix of \(G\). \(L \triangleq L \otimes I_{\tau_n}\). Denote the total number of edges within \(G\) as \(m\). Define the oriented incidence matrix of \(G\) denoted by \(H \in \mathbb{R}^{m \times N_u}\) such that its entry at the \(k\)-th row and the \(r\)-th column is 1 if edge \(k\) is an incoming edge to node \(r\); -1 if edge \(k\) is an outgoing edge to node \(r\); and 0 elsewhere. Note that for undirected graphs, the direction for each edge could be arbitrary. Denote \(\bar{H} \triangleq H \otimes I_{\tau_n}\).
Algorithm 1: 2D Speed-Torque PDF Generation

Input: $N_{\tau}, N_{\omega}, \xi, N_s = 0, \omega_{\text{max}}, \tau_{\text{max}}$

1. $q \leftarrow \text{zeros}(N_{\tau}, N_{\omega})$ // Initialize the PDF
2. $v_{\omega} \leftarrow \text{linespace}(0, \omega_{\text{max}}, N_{\omega} + 1)$
3. $v_{\tau} \leftarrow \text{linespace}(0, \tau_{\text{max}}, N_{\tau} + 1)$
4. for $i_{\omega} = 1$ to $N_{\omega}$ do
5.   $\omega_1 \leftarrow v_{\omega}[i_{\omega}], \omega_2 \leftarrow v_{\omega}[i_{\omega} + 1]$
6.   for $i_{\tau} = 1$ to $N_{\tau}$ do
7.     $\tau_1 \leftarrow v_{\tau}[i_{\tau}], \tau_2 \leftarrow v_{\tau}[i_{\tau} + 1], c \leftarrow 0$
8.     for each datapoint in $\xi$ do
9.         $\omega, \tau_m \leftarrow \text{parse}(\text{datapoint})$
10.        if $\tau_m \in [\tau_1, \tau_2]$ and $\omega_m \in [\omega_1, \omega_2]$ then
11.           $c \leftarrow c + 1, N_s \leftarrow N_s + 1$
12.        if $i_{\tau} == N_{\tau}$ and $\tau_m \geq \tau_2$ and $\omega_m \in [\omega_1, \omega_2]$ then
13.           $c \leftarrow c + 1, N_s \leftarrow N_s + 1$
14.     end for
15.   end for
16. end for
17. $q \leftarrow q/N_s$ // Normalize the PDF

The SPMSM design variables for manipulator-$i$’s $j$-th motor are denoted by $l_{i,j}, r_{\text{ro},i,j}, r_{\text{so},i,j}, h_{m,i,j}, h_{s,y,i,j}, w_{\text{tooth},i,j}, b_{0,i,j}$. And further denote

$$\beta_{i,j} \equiv \{l_{i,j}, r_{\text{ro},i,j}, r_{\text{so},i,j}, h_{m,i,j}, h_{s,y,i,j}, w_{\text{tooth},i,j}, b_{0,i,j}\},$$

where $\beta_i$ indicates all the motor design parameters of manipulator $i$. Similar as (6), the complete dynamics for manipulator $i$ are written as

$$\dot{x}_i = f_i(x_i(t), u_i(t), \beta_i),$$

where $x_i \equiv \{\theta_i, \dot{\theta}_i, x_{m,i}\} \in \mathbb{R}^{4n}$ and $u_i \equiv u_{m,i} \in \mathbb{R}^{2n}$.

Given a task, a trajectory planning optimization can return the optimal trajectories of states and controls that align with the task requirements:

$$\begin{align*}
\min_{x_i(t), u_i(t), t_{f,i}} & \quad J_i(x_i(t), u_i(t), t_{f,i}) \\
\text{s.t.} & \quad x_{i}(t) = f_i(x_i(t), u_i(t), \beta_i), \quad \forall t \in [0, t_{f,i}] \\
& \quad \text{constraints (3), (5)}, \quad \forall t \\
& \quad \theta_i(t_{f,i}) = \theta_{\text{des},i}, \quad \dot{\theta}_{i}(t_{f,i}) = 0,
\end{align*}$$

where $t_{f,i} > 0$ denotes the final time for trajectory planning, which could either be a decision variable or a fixed prescribed parameter; $\theta_{\text{des},i}$ denotes a prescribed desired final position. Note that (9) adopts the most general form, which represents time-optimal trajectory planning, trajectory tracking, or energy-optimal trajectory planning. For the latter two cases, $t_{f,i}$ is a fixed prescribed parameter. Given a particular value of $\beta_i$, the optimal states $x^*_i(t)$, controls $u^*_i(t)$, and final time $t^*_{f,i}$ (if applicable) optimize the cost function $J_i$. For notational simplicity, let $\xi_i \equiv \{x^*_i(t), u^*_i(t), \forall t, t^*_{f,i}\}$ denote the optimal trajectory of manipulator $i$ given the task.

Then similar to (7), the distributed manipulator co-design problem can be rewritten as:

$$\begin{align*}
\min_{\beta_1, \ldots, \beta_N} & \quad \sum_{i=1}^{N} \ell_i(\xi_i, \beta_i) \\
\text{s.t.} & \quad \beta_1 = \cdots = \beta_{N_s}, \\
& \quad \xi_i \text{ obtained from (9) given } \beta_i, \\
& \quad h_{d,i}(\beta_i) \leq 0, \\
& \quad h_{o,i}(\beta_i, \tau_{m,i}, \omega_i) \leq 0, \\
& \quad \forall (\tau_{m,i}, \omega_i) \in A_i, \forall i \in \mathcal{V},
\end{align*}$$

where $\ell_i(\cdot)$ is a local loss function that measures the total operational efficiency loss of manipulator-$i$’s $n$ motors, i.e.

$$(1 - \eta(\tau_{m,i}, \omega_i; \beta_{i,j}))d\omega_i dt_{m,i,j}.$$ 

Combining ADMM and ALM, this section proposes a distributed constrained optimization algorithm to solve (10) iteratively. Originally, the augmented Lagrangian of (10) is not separable for each $\beta_i$ and hence the gradient descent of this Lagrangian w.r.t. $\beta$ cannot be distributed to each $\beta_i$. Thus, an equivalent augmented Lagrangian is presented such that the update rule is distributed for $\beta_i$. First, an assumption and a theorem are introduced.

**Assumption 1.** The undirected graph $\mathcal{G}$ is connected.

**Theorem 1.** Let Assumption 1 hold. Denote $\hat{\beta} \equiv \text{col}(\beta_1, \ldots, \beta_{N_s}) \in \mathbb{R}^{7nN_s}$. The consensus constraint (10b) is equivalent to $\mathcal{L}\hat{\beta} = 0$ and further $\mathcal{H}\hat{\beta} = 0$.

**Proof.** By Assumption 1, the graph $\mathcal{G}$ is connected. Then the consensus constraint $\beta_1 = \cdots = \beta_{N_s}$ holds if and only if $\mathcal{L}\hat{\beta} = 0$. Given $\mathcal{L} = H^T H$, $\mathcal{L} \equiv H^T H \otimes I_{7n} \equiv (H^T \otimes I_{7n})(H \otimes I_{7n}) \equiv H^T H$. Then the consensus constraint holds if and only if $H^T \hat{H} \hat{\beta} = 0$. Apparently $H^T \hat{H} \hat{\beta} = 0 \Rightarrow H^T H \hat{\beta} = 0$. Since $\mathcal{G}$ is connected, from [26, Theorem 8.3.1], there is one connected component of $\mathcal{G}$, and hence ker $H = 1$. Thus, ker $H^T = 0$ and ker $H = 0$. Because the kernel’s definition, $H^T H \hat{\beta} = 0 \Rightarrow \hat{\beta} = 0$. Therefore, the consensus constraint holds $\Leftrightarrow \mathcal{L}\hat{\beta} = 0 \Leftrightarrow \mathcal{H}\hat{\beta} = 0$. 

Define $g_i(\beta_i, \tau_{m,i}, \omega_i) \equiv \max(h_{o,i}(\beta_i, \tau_{m,i}, \omega_i) , 0)^2$, where max($\cdot$) and square are applied element-wise. Then the inequality constraint (10e) can be converted to an equality constraint, i.e. $g_i(\beta_i, \tau_{m,i}, \omega_i) = 0$. Together with
Theorem 1, (10) is equivalent to an equality-constrained optimization:

\[
\begin{align*}
\min_{\hat{\beta} \in B \subset \mathbb{R}^{7nN_a}} & \sum_{i=1}^{N_a} \ell_i(\xi_i, \beta_i) \\
\text{s.t.} & \quad \check{\mathcal{L}}\hat{\beta} = 0, \\
& \quad g_i(\beta_i, \tau_{m,i}, \omega_i) = 0,
\end{align*}
\]  

where \( B \) represents the feasible set among all agents, and \( B_i \subset \mathbb{R}^{7n} \) represents agent-\( i \)'s feasible set constrained by \( h_{d,i}(\beta_i) \leq 0 \) in (10d). Introduce the Lagrangian multipliers \( \lambda_i \) and \( \lambda_L \) associated with (12b) and (12c), respectively. \( \lambda_L \) is given by \( \text{col}\{\lambda_{L,1}, \ldots, \lambda_{L,N_a}\} \in \mathbb{R}^{7nN_a} \), where \( \lambda_{L,i} \in \mathbb{R}^{7n} \); with the same dimension, denote \( \lambda \) as \( \text{col}\{\lambda_1, \ldots, \lambda_{N_a}\} \). Similar to [27], with \( \check{\mathcal{L}}\hat{\beta} = 0 \Leftrightarrow \tilde{H}\hat{\beta} = 0 \) from Theorem 1, the augmented Lagrangian associated with (12) is equivalent to

\[
L(\hat{\beta}, \lambda_L, \lambda) \equiv \lambda_L^T \check{\mathcal{L}}\hat{\beta} + \frac{\rho_2}{2} \|	ilde{H}\hat{\beta}\|^2 + \sum_{i=1}^{N_a} (\ell_i(\xi_i, \beta_i) + \lambda_L^T g_i(\beta_i) + \frac{\rho_2}{2} \|g_i(\lambda\|^2),
\]

where \( \rho_1, \rho_2 > 0 \) are arbitrary constants. Denote \( k \in \mathbb{Z}_+ \) as the iteration index. To compute the optimal \( \hat{\beta} \), the gradient descent in \( \lambda \) and gradient ascent in \( \lambda_L \) and \( \lambda \) can be applied in an ALM fashion [28]. Thus, with the fact that \( \check{\mathcal{L}}_i = \check{\mathcal{L}} \), the iterative update rule is given by

\[
\begin{align*}
\hat{\beta}(k+1) &= \hat{\beta}(k) - \alpha (L\lambda_L(k) + \rho_1 \check{\mathcal{L}} \hat{\beta}(k) + (*)), \\
\lambda_L(k+1) &= \lambda_L(k) + \alpha \lambda_i \check{\mathcal{L}} \hat{\beta}(k+1), \\
\lambda_i(k+1) &= \lambda_i(k) + \alpha g_i(\beta_i(k+1), \cdot),
\end{align*}
\]

where (*) indicates the partial derivative of the summation term in (13) w.r.t. \( \hat{\beta} \); \( \alpha > 0 \) is a step size. Then the update (14) is naturally distributed and can be decomposed as:

\[
\begin{align*}
\tilde{\beta}_i(k+1) &= \beta_i(k) - \alpha \sum_{r \in \mathcal{N}_r(i)} (\lambda_L(\beta_r(k) - \lambda_L(\beta_r(k))) \\
& \quad + \rho_1 \beta_i(k) - \rho_2 \beta_i(k)) - \alpha \frac{\partial \ell_i(\xi_i, \beta_i)}{\partial \beta_i}^T \lambda_i - \alpha \rho_2 \frac{\partial g_i(\lambda\|^2}{\partial \beta_i}^T g_i(\cdot), \\
\beta_i(k+1) &= \arg \min_{x \in B_i} \|\tilde{\beta}_i(k+1) - x\|_2, \\
\lambda_L,i(k+1) &= \lambda_L,i(k) + \alpha \sum_{r \in \mathcal{N}_r(i)} (\beta_r(k+1) - \beta_r(k+1)), \\
\lambda_i(k+1) &= \lambda_i(k) + \alpha g_i(\beta_i(k+1), \cdot),
\end{align*}
\]

where \( \frac{\partial \ell_i(\xi_i, \beta_i)}{\partial \beta_i} \) and \( \frac{\partial g_i(\cdot)}{\partial \beta_i} \) are given by the definition in (11), (10) and (7). Based on the update rule (15), the distributed co-design algorithm is summarized in Algorithm 2, where the content within parfor is executed by each agent distributedly.

**Remark 2.** The motor design constraints (10d) are enforced as a projection instead of inequality constraints because the projection ensures the feasibility of trajectory planning given updated parameters \( \beta_i(k+1) \) for every iteration \( k \).

**Algorithm 2: Distributed Co-Design Algorithm**

- **Input:** \( \rho_1, \rho_2, N_\gamma, N_\omega, N_k \in \mathbb{Z}_+, \alpha > 0 \)
- **Initialization:** \( k \leftarrow 0; \lambda_{L,i}(k) = \lambda_L(k) = 0, \) Initialize \( \beta_i(k), \forall i \)
- Each agent \( i \) initializes \( \xi_i(k) \) by (9) given \( \beta_i(k) \)
- Each agent \( i \) obtains \( \beta_i(k) \) from neighbors \( r \in N_i \)
- while \( k < N_k \) do
  - parfor agent \( i = 1 \) to \( N_a \) do
    - Obtain \( q_{i,j}(\cdot), \forall j = 1, \ldots, n \) by Algorithm 1
    - Obtain \( \lambda_{L,i}(r) \) from neighbors \( r \in N_i \)
    - \( \beta_i(k+1) \leftarrow \) update by (15a), (15b)
    - \( \xi_i(k+1) \leftarrow \) solve (9) given \( \beta_i(k+1) \)
    - Obtain \( \beta_i(k+1) \) from neighbors \( r \in N_i \)
    - \( \lambda_{L,i}(k+1), \lambda_i(k+1) \leftarrow \) update by (15)
  - \( k \leftarrow k + 1 \)

V. **Numerical Results**

This section presents a numerical simulation of the distributed co-design of 6-DOF manipulators. Suppose there is a complete graph with 8 agents, where each agent includes a trajectory planner (9) with its task specification. For agent-\( i \), the cost function \( J_i \) of (9a) represents the copper loss of all \( n \) motors and is given by \( J_i = \int_0^{t_f} \sum_{i,j} (i_{d,i}(t)^2 + i_{q,j}(t)^2)dt \), where \( t_{f,i} \) varies from 2.0 to 3.5 s. For simplicity, each agent’s task is to move a solid iron ball to a desired final position. Each agent’s dynamics (9b) can be computed by Remark 1 with the ball mass varying from 0.5 to 1.2 kg. The initial state \( x(0) \) in (9c) is given by \( \text{col}\{-0.74, 0.6, 0.96, 2.02, 0.84, -2.19\} \) rad. The desired final position \( \theta_0 = \text{col}\{-3.14, 0.59, 1.79, 0, -0.81, 0\} \) rad. The joint constraints (5) are given by \( \tilde{\theta} : = \text{col}\{2\pi, 0.6\pi, 0.6\pi, 2\pi, 0.6\pi, 2\pi\} = -\tilde{\theta} \) and \( \theta = 100\pi \cdot 1_n = -\tilde{\theta} \) for all agents and joints. The trajectory planning problem (9) is solved by the direct collocation method and referred to [25, Section III.B] for details. Let \( \rho_1 = 0.75, \rho_2 = 10, \alpha = 0.08, N_\gamma = N_\omega = 20 \).

The distributed co-design algorithm, i.e. Algorithm 2, is utilized to update \( \beta_i \), where the initial design \( \beta_i(0) \) is obtained empirically given each task. The global average of motor loss, i.e. \( \frac{1}{n} \sum_{i=1}^{n} \ell_i(\beta_i(k))/\sum_{n} \ell_i(\cdot) \cdot 100\% \), over iterations is shown in Fig. 3. It shows that the average motor loss decreases by 4% after 60 iterations. Fig. 3 also shows the consensus error, i.e. \( \sum_{i,j} \|\tilde{\beta}_i(k) - \tilde{\beta}_j(k)\|_2 \), converges to zero. It converges three times within 60 iterations because there exist multiple consensual \( \beta_i \) that satisfy all the constraints. The gradient of \( \sum_{i,j} \ell_i(\cdot) \) and of the constraints drive \( \beta_i \) to another consensual point with a lower loss. The global average of loss does not monotonically decrease because of the non-convexity of (12). And there might not be a global optimum that is also optimal for every individual loss function. Hence, the global average increases when the consensus error decreases. Ultimately, the proposed algorithm drives \( \beta_i \) to a local and consensual optimum. Fig. 4 further validates that the distributed co-design improves
the motor operation efficiency. The black circles represent the motor operation points from the optimal trajectory given a particular design, where the circle radius indicates the probability of this operation point. The numerical results altogether verify the effectiveness of the proposed distributed co-design algorithm.

VI. CONCLUSION

This paper investigates a motion and motor co-design problem for robotic manipulators given multiple tasks. Existing co-design methodologies typically solve trajectory planning given each task individually and then optimize motor efficiency given these trajectories. With the number of tasks increasing, the computational burden grows significantly. To reduce computational burden and improve scalability as the number of tasks grows, this paper introduces a distributed co-design framework to handle the co-design process for all tasks in a distributed fashion. This paper further presents a consensus-based distributed optimization algorithm, which secures a unified set of design parameters for all tasks ultimately such that the total average of motor operation efficiency is optimized and the design constraints are satisfied across all tasks and motors. The distributed manner reduces the computational load by allowing each agent to solve co-design optimization solely for its designated task. A numerical simulation further verifies the proposed algorithm.

Future improvements include a thorough theoretical analysis of the proposed algorithm and developing a more efficient gradient descent algorithm, such as momentum-based gradient descent, etc. In addition, this paper focuses on optimizing motor efficiency based on open-loop optimal trajectories from trajectory planners. In practice, one also needs to design a closed-loop tracking controller to track the optimal trajectory for each task. Thus, how to co-design the motors, motions, and closed-loop controller of robotic manipulators given multiple tasks is a valuable direction.

REFERENCES

For a rectangular tooth cross-section, one can compute the slot width as $b_{ss,j} = \frac{A_{slot,j}}{h_{ss,j}}$, where $A_{slot,j}$ is the slot area, i.e.

$$A_{slot,j} = \frac{\pi((r_{cm,j} - h_{av,j})^2 - (r_{ro,j} + \delta + h_{tip,j})^2)}{Q} - w_{tooth,j}h_{ss,j}. \tag{17}$$

The cross-section area of the stator core is given by

$$A_{so,j} = \pi r_{so,j}^2 - \pi (r_{ro,j} + \delta)^2 - Q(A_{slot,j} + b_0h_{tip}). \tag{18}$$

Thus the volume of the stator core is given by

$$V_j = A_{so,j}l_j. \tag{19}$$

The copper area is given by $A_{cu,j} = A_{slot,j}f_1$. For concentrated windings and assuming one winding is a complete turn around a tooth, the area of a single coil is given by $A_{coil,j} = A_{cu,j}/(2n_s)$. The minimal wire diameter is

$$D_{wire,j} = \frac{4A_{coil,j}}{\pi}. \tag{20}$$

The arc span per slot can be determined by $\tau_{s,j} = 2\pi(r_{ro,j} + \delta)/Q$. The average length of the coil end-winding $l_{end,av,j}$ and the total coil length $l_{coil,j}$ are given by

$$l_{end,av,j} = (w_{tooth,j}(2 - \pi/2) + \pi r_{s,j}/2)/2, \quad l_{coil,j} = 2l_j + 2l_{end,av,j}.$$ 

Then the weight of the stator and rotor are given by:

$$m_{rotor,j} = \rho_{iron}\pi r_{ro,j}^2l_j, \quad m_{stator,j} = \rho_{iron}\pi r_{so,j}^2l_j - \rho_{iron}\pi(r_{ro,j} + \delta)^2l_j - \rho_{iron}A_{slot,j}l_jQ + \rho_{cu}A_{coil,j}n_sQ. \tag{21}$$

Without loss of generality, define the x-axis as the central axis of each rotor or stator, i.e. x-axis coincides with the axial length of core $l_j$; consequently, define the y-axis and z-axis by following the right-hand rule and the two axes indicate the central radius. All three axes originate at the centroid of the rotor or stator, i.e. the center of the axial length $l_j$. Since each rotor is a solid cylinder, the moment of inertia about three principal axes of each rotor is given by:

$$I_{xx,j} = \frac{1}{2}\rho_{iron}\pi r_{ro,j}^4l_j = \frac{1}{4}m_{rotor,j}l_j^2, \quad I_{yy,j} = I_{zz,j} = \frac{1}{4}\rho_{iron}\pi r_{ro,j}^2(3r_{ro,j}^2 + l_j^2).$$

To simplify the inertia calculation for stators, each stator is simplified as a hollow cylinder with outer radius $r_{so,j}$ and inner radius $r_{ro,j} + \delta$. Then the moment of inertia about three principal axes of each stator is given by:

$$I_{xx,j} = \frac{1}{4}m_{stator,j}(r_{so,j}^2 + (r_{ro,j} + \delta)^2), \quad I_{yy,j} = I_{zz,j} = \frac{1}{12}m_{stator,j}(3r_{so,j}^2 + (r_{ro,j} + \delta)^2 + l_j^2).$$
2) Resistance: The resistance per tooth is given by \( R_{1,j} = (n_s^2 R_{c,coil,j})/(A_{slot,j} f_j) \). Using this, the phase resistance can be calculated as
\[
R_j = q_1 R_{1,j} / C_p^2.
\]

3) Permeance: The permeance of the magnetic path across the air gap and the slot opening, denoted by \( p_{g,j} \) and \( p_{so,j} \), are given by:
\[
p_{g,j} = \frac{2\pi r_{p,j} m_{l,j}}{q + \pi m_{l,j} / P_r}, \quad p_{so,j} = \frac{\mu_0 h_{so,j}}{h_{so,j}}.
\]
The permeance of the magnetic path that curves from tip to tip is given by \( p_{ht,j} = \frac{\mu_0 (\phi + h_{so,j})/2 + h_{so,j}}{\pi (\phi + h_{so,j})} \).

4) Inductance: For an SPMSM, its d-axis and q-axis inductance are equivalent to each other and given by
\[
L_{d,j} = L_{q,j} = q_1 n_s^2 L_1 / C_p^2,
\]
where \( L_{1,j} \) is the inductance per turn and per tooth, given by \( L_{1,j} = p_{g,j} + 3p_{so,j} + 3p_{ht,j} \).

5) Flux: To proceed with the calculation, it is necessary to determine Carter’s coefficient denoted by \( k_{C,j} \), given by
\[
k_{C,j} = \frac{t_{pitch,j}}{t_{pitch,j} - \gamma_j}, \quad \gamma_j = \left( \frac{b_{so,j}}{\delta_j} \right)^2, \quad t_{pitch,j} = \frac{2\pi r_{p,j}}{q}.
\]
Then, the magnetic flux density across the gap is given by
\[
\Phi_{m,j} = B_{h,j} \frac{h_{so,j}}{n_{so,j} / \mu_0 + k_{C,j}}.
\]
The flux density corresponding to the first harmonics can be calculated as \( B_{h,1,j} = 4B_{h,j}/\pi \). Consequently, the flux per tooth per single turn is given by
\[
\Phi_{1,j} = B_{h,1,j} l_j 2\pi r_{p,j} / Q.
\]
In the absence of skewness, the flux linkage is given by
\[
\Phi_{m,j} = k_w n_s \Phi_{1,j} q_1 / C_p,
\]
where \( k_w = k_p l_d \) denotes the winding factor, and
\[
k_{w} = \sin(\pi p / Q), \quad k_d = \frac{\sin(\pi / 6)}{\sin(\pi / 6) + \sin(\pi / 6)}.
\]

B. SPMSM Torque & Efficiency Modeling

This subsection introduces the modeling of SPMSM torque, maximum torque, and efficiency. The constant parameters used here are:

- Operating temperature \( T_{op} = 80 \textdegree C \)
- Ambient temperature \( T_{amb} = 20 \textdegree C \)
- Motor’s q-axis voltage upper bound \( \bar{v}_{q,j} = 100 \text{ V} \)
- Motor’s q-axis current upper bound \( \bar{i}_{q,j} = 3 \text{ A} \)
- Maximum motor power \( P_{max,j} = 600 \text{ W} \)

1) Torque: First, the Hysteresis loss and eddy current loss are given by Steinmetz’s equation, i.e.,
\[
P_{hyst,j} = k_{hyst} \left( \frac{\bar{v}_{q,j}}{2} \right)^2 B_j^{1.6} V_j,
\]
\[
P_{eddy,j} = k_{eddy} \left( \frac{\bar{v}_{q,j}}{2} \right)^2 B_j^2 V_j,
\]
where \( k_{hyst} = 130 \text{ Ws}^{-1} \text{m}^{-3} \) and \( k_{eddy} = 1.1 \text{ Ws}^{-2} \text{m}^{-3} \) are the coefficients, estimated by finite element analysis (FEA) simulation of a base design motor. \( V_j \) is the volume of motor-j’s stator core and defined in (19); \( B_j \) is the average flux density at the tooth, calculated as follows:
\[
B_j = \frac{u_{n}(q_j/C_p)w_{m,j} / \omega_{j,0}}{\Phi_j}
\]
\[
\Phi_j = \sqrt{(\Phi_{m,j} + L_{d,j} i_{d,j})^2 + (L_{q,j} i_{q,j})^2}
\]
where \( \Phi_j \) indicates the total flux generated by both permanent magnets and coils. The loss torque is given by
\[
\tau_{hyst,j} = P_{hyst,j} / |\omega_j|, \quad \tau_{eddy,j} = P_{eddy,j} / |\omega_j|
\]
Then the motor torque \( \tau_{m,j} \) of motor j is given by
\[
\tau_{m,j} = \frac{1.5p \Phi_{m,j} i_{q,j}}{q_1} + (L_{d,j} - L_{q,j}) i_{d,j} i_{q,j},
\]
\[
\tau_{m,j} = \hat{\tau}_{m,j} - \text{sign}(\hat{\tau}_{m,j})(\tau_{hyst,j} + \tau_{eddy,j}),
\]
where \( \text{sign}() \) denotes the signum function. Then the joint torque \( \tau_j \) at joint j is given by
\[
\omega_j = \hat{\theta}_j Z_j, \quad \tau_j = \tau_{m,j} Z_j,
\]
where \( \hat{\theta}_j \) is the angular velocity of joint j.

Given an arbitrary optimal trajectory of joint-j’s position and velocity and motor-j’s current and voltage from a trajectory planner (9), the motor torque and velocity at each time instance can be calculated by (31) and (32), respectively. Then the maximum (magnitude) motor torque \( \tau_{max,j} \) and the maximum (magnitude) motor velocity \( \omega_{max,j} \) of motor j can be obtained directly from the trajectory of motor-j’s torque and velocity.

2) Efficiency: The efficiency \( \eta_j(\tau_{m,j}, \omega_j, \beta_j) \) is determined by motor-j’s torque \( \tau_{m,j} \) and motor’s velocity \( \omega_j \) under the motor design parameter \( \beta_j \in \mathbb{R} \). The efficiency \( \eta_j(\tau_{m,j}, \omega_j, \beta_j) \) is given by:
\[
\eta_j = \frac{P_{in,j} - P_{cu,j} - \hat{P}_{hyst,j} - \hat{P}_{eddy,j}}{P_{in,j}},
\]
where \( P_{in,j}, P_{cu,j} \) are the input power and copper losses power of motor j, respectively. \( P_{in,j}, P_{cu,j} \) are given by
\[
P_{in,j} = u_{d,j} i_{d,j} + u_{q,j} i_{q,j}, \quad P_{cu,j} = R_T (i_{d,j}^2 + i_{q,j}^2),
\]
\[
R_T = R_j T/T_{amb}, \quad T = T_{op} P_{in,j} / P_{max,j} + T_{amb},
\]
where \( R_j \) is defined in (22); \( T \) is the scaled operating temperature; (34b) calculates the copper resistance given scaled temperature by linear extrapolation. \( \hat{P}_{hyst,j}, \hat{P}_{eddy,j} \), currents and voltages are calculated as follows:
\[
\Phi_{max,j} = (\pi \bar{v}_{q,j} - R_j T_{amb} / L_{d,j}) / |\omega_j|, \quad \Phi_{max,j} = (\pi \bar{v}_{q,j} - R_j T_{amb} / L_{d,j}) / |\omega_j|
\]
\[
\hat{\Phi}_j = \text{min}(\Phi_{max,j}, \Phi_{m,j}),
\]
\[
i_{d,j} = (\hat{\Phi}_j - \Phi_{m,j}) / L_{d,j}, \quad i_{q,j} = (|\tau_{m,j}| + \hat{\tau}_{hyst,j} + \hat{\tau}_{eddy,j}) / (1.5p \Phi_{m,j}),
\]
\[
u_{d,j} = R_j i_{d,j} - p |\omega_j| i_{q,j} L_{d,j}, \quad \nu_{q,j} = R_j i_{q,j} + p |\omega_j| \hat{\Phi}_j,
\]
where \( \Phi_{max,j} \) is the maximum flux given motor velocity and maximum voltage and current; (35b) ensures flux weakening; \( \hat{P}_{hyst,j}, \hat{P}_{eddy,j}, \hat{\tau}_{hyst,j}, \hat{\tau}_{eddy,j} \) are calculated by following (28), (29a), and (30), where \( \Phi_j \) in (29a) is replaced by \( \hat{\Phi}_j \) in (35b).