Received April 27, 2022, accepted May 14, 2022, date of publication May 23, 2022, date of current version May 27, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3177209

Parameter Identification of a Robot Arm Manipulator Based on a Convolutional Neural Network

CARLOS LEOPOLDO CARREÓN-DÍAZ DE LEÓN[1], SERGIO VERGARA-LIMÓN[2], MARÍA AURORA D. VARGAS-TREVIÑO[2], JESÚS LÓPEZ-GÓMEZ[3], JUAN MANUEL GONZALEZ-CALLEROS[1], DANIEL MARCELO GONZÁLEZ-ARRIAGA[1], AND MARCIANO VARGAS-TREVIÑO[4]

[1]Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla (FCC-BUAP), Puebla 72570, Mexico
[2]Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla (FCE-BUAP), Puebla 72570, Mexico
[3]División Académica de Ingeniería y Arquitectura, Universidad Juárez Autónoma de Tabasco (DAIA-UJAT), Tabasco 86040, Mexico
[4]Escuela de Sistemas Biológicos e Innovación Tecnológica, Universidad Autónoma Benito Juárez de Oaxaca (ESBIT-UABJO), Oaxaca 68120, Mexico

Corresponding author: Carlos Leopoldo Carreón-Díaz de León (carlos.carreond@alumno.buap.mx)

This work was supported in part by the Consejo Nacional de Ciencia y Tecnología (CONACyT) by the Postgraduate Scholarship.

ABSTRACT Dynamic parameters are crucial in designing robotics systems because they reflect an actual robot. Conventional identification methods require that the robot execute the optimal motion; however, they spend time trying all possible trajectories in the robot. This article shows the identification of a robot arm of 2 degree-of-freedom with an algorithm based on a convolutional neural network (CNN) and their dynamic model. The proposal consists of a CNN that uses an image construct with a proposed conversion technique and the robot signals. The algorithm gets the parametric residuals from this signal image to find the parameters without trajectory optimization. An embedded system on a Field Programmable Gate Array (FPGA) has the classical controller Proportional-Derivative to execute a predefined trajectory for the identification. The identified parameters and the predefined motion rebuild the torque with the dynamic model. A proposed evaluation metric based on the discrete cosine transform evaluates the similarity of the actual and reconstructed torques. Four numeric tests verify the algorithm by torques created with the dynamic model, the predefined trajectory, and a parameter set. The similarity of numeric torques and their rebuilding overcomes 97.38%, and the experimental and rebuilt torque with the identified parameters is over 93.55%. The proposed algorithm is compared with least-squares, and the results show that the proposed method provides better identification of the experimental robot.

INDEX TERMS CNN, dynamic parameters, parameter identification, robotics, signals.

I. INTRODUCTION

The parametric identification is a research line that allows the dimensionality of a mathematical model of a determined robot to achieve the emulation by computer of complex applications like the position control [1] or the collision detection of a manipulator with the dynamic parameters [2]. The identification of a robot consists of finding the number of parameters associated with the ordinary differential equation. Usually, these parameters are linearly independent, and the measures of position, velocity, acceleration, and torque are necessary to determine the parameters. In [3], the basic steps to optimize a trajectory are described to obtain the best excitation signals for parameter identification. The method is validated with a 3 degree of freedom (DoF) robot in that work, and the identification is carried out with maximum likelihood. The survey of [4] shows that least squares (LS), Kalman filter (KF), and optimization methods like genetic algorithms and particle filters are commonly used for parameter identification. In [5], a Matlab identification procedure with LS for students is implemented. The proposed method is illustrated with a 2 DoF robot manipulator and

The associate editor coordinating the review of this manuscript and approving it for publication was Guilin Yang.

a Sensoray DAQ. Neural networks (NN) have been studied and applied in many scientific and industrial areas. The convolutional neural network (CNN) architecture has many advantages due to its area-based convolutions layers: The typical but not limited applications are image classification, object detection, segmentation, and recognition, as the survey [6] shows. The neural networks have an essential role in robotics applications, such as the work presented in [7], where is studied a NN with a partial connection between the input and hidden layers to the position control of a robot. In [8], there is a comparative study for a convolutional neural network (CNN) and a feed-forward neural network (FFNN) to identify the movement of three mechanical systems. This work shows that the CNN has a better performance in handling the input signals than the FFNN. In [9], an FFNN is used to identify the applied force in a robot tool for medical surgery. However, in [10], it is presented the same application, but instead of the FFNN, a deep convolutional neural network (DCNN) is used. These articles show that DCNN identifies the force of the tool more accurately than the FFNN.

A. RELATED WORK

The identification of the KUKA KR R700 robot by the pseudo inverse observation matrix is performed in [11]. It is used an FFNN to estimate the friction phenomena with a better experimental result than the mathematical model of friction. The external force is estimated by the proposed disturbance observer Kalman filter. The acceleration estimation is avoided using mathematic identities of the dynamic models in [12] to identify the parameters of two robots of 2 DoF and 5 DoF. In [13], a convex optimization method is used to identify the parameters of the proposed dynamic model of the Stäubli RX-160 robot of 6 DoF. The Coulomb friction is modeled by an arctangent function because the discontinuities created by the sign function commonly used affect the identification procedure. A radial basis neural network (RBNN) is used to estimate the stiffness and inertia parameters of a 1 DoF robot in [14] to be used in adaptive position control. The proposed system is illustrated by simulation, and the results show that the position error is reduced from 0.2 rad with a proportional derivative (PD) control to $+-0.05$ rad. In [15], a long short term memory (LSTM) network is used to develop the semiparametric identification of a UR5 robot of 6 DoF. The inputs of the proposed system are the time sequences of position, velocity, and acceleration, while the output is the torque. The results exhibit that the torque estimation with the semiparametric method is more accurate than the LSTM without the inertia and gravity model of the robot. The calibration of a strain-wave transmission system of a UR5e robot of 6 DoF is executed in [16] to improve the weighted LS (WLS) parametric identification. The robot uses two sensors for each joint to compensate for the flexibility of the transmission system. The results are compared with a torque generated by computer-assisted design (CAD) software.

The global optimization is used in [17] to identify the parameters of a KUKA LBR iiwa 14 R820. The results confirm that the identified parameters can accurately predict the experimental torque. An uncertainty compensation system of the torque based on an LSTM is proposed in [18] to enhance the results of the parametric identification by LS of a UR5 robot of 6 DoF. The results show that the LSTM has a better compensation than an FFNN. In [19], a new friction model is developed that consists of the Coulomb friction in two directions and the viscous friction by a polynomial equation. The friction model is validated by the parametric identification of a ROKAE BX7 robot of 6 DoF using WLS. The results confirm that the proposed friction model allows a better torque estimation than the classic friction model. In [20], a collision detection system is proposed that includes the parameter identification by LS of an Efor ER3A robot of 6 DoF considering the uncertainties. The identification is conducted in two steps: the gravitational torque and the friction terms, while the remaining terms are recovered by the machine learning method Lasso. A whale optimization algorithm with a genetic algorithm (WOA-GA) is used to identify the parameters of a Mitsubishi RV-4FL of 6 DoF in [21]. The proposed friction model is a function of the velocity and the acceleration. It is named the centrosymmetric friction model. The results prove that the WOA-GA has a coincidence of 93.83% compared to only the whale optimization algorithm (WOA) 92.02% and only the genetic algorithm (GA) 93.27%.

The Franka Emika Panda robot of 8 DoF is used to identify their feasible parameters in [22]. The parameters identified are the mass, inertia, and center of mass using an optimization method with physics restriction. In [23], the identification of an industrial robot of 6 DoF by LS is implemented. The dynamic model considers that the joints are flexible, and before making the identification, the stiffness is previously determined by experimentation. In consequence, it is not necessary to have a dual encoder sensor. The results illustrated by simulation show that the precision is improved with this adjustment. The external force of a KUKA KR 6 R700 robot of 6 DoF is determined by the dynamic model in [24]. It is used WLS to identify the inertia and friction parameters in combination with a generalized Maxwell slip model for the torque changes. The methodology can estimate the external force accurately in comparison with an external sensor. An RBNN is used to identify two dynamic systems in [25]. It is implemented an adaptive position control that uses the descending gradient method. The simulation results determine that the RBNN enhances the performance of the control to 0.0023 of the mean square error. The filtered dynamic model of a robot with flexible joints to the parameter identification by LS is used in [26], and the acceleration measures are not needed. The external contact force of a Kinova Jaco2 robot of 6 DoF is developed in [27]. The methodology identifies the dynamic parameters by LS and compensates the errors with a multi-layer perceptron (MLP). With a disturbance KF, the external force is estimated. Conventional parameter identification of an industrial robot of 6 DoF by LS is developed in [28].

The main contribution of this research is the parametric identification of a two-degree-freedom robotic arm. An identification algorithm based on a convolutional neural network and the dynamic model of the robot identifies the parameters of the dynamic model. The input of the proposed CNN is an image generated with the subsampled robot signals made by a spectral frequency of the discrete cosine transform technique. The parameter identification algorithm is tested in a real 2 DoF robot arm. The position control system and the developing data acquisition system are designed into an FPGA. The parameters are validated with two pseudo aleatory filtered test trajectories, a cartesian circle, and a profile trajectory. One of the features of the proposed algorithm is that training data is generated by replacing position, speed, and acceleration to obtain the torque numerically, avoiding implementing simulations that take longer. The proposed method does not require trajectory optimization: This traditional step requires testing every new trajectory in the experimental robot and takes more time than the conventional identification methods. A comparison between the proposed algorithm and LS is implemented to show that the proposed method overcomes LS.

TABLE 1. State of the art comparison.

Reference	Method	Optim. Trajectory	Inertial parameters	Friction parameters	Torque estimation
[11]	Pseudo-inv.		X		
[12], [20], [23], [26], [27], [28]	LS	X	X X	X	
[16], [19], [24]	WLS	X	X	X	
[13], [17], [21], [22]	Optim. methods	X	X	X	
[15]	LSTM		X		X
[18]	LS-LSTM	X	X	X	X
[14], [25]	RBNN		X		X
Our method	CNN extraction		X	X	

Table 1 compares the related work and the proposed identification methodology. The works based on LS, WLS, and some optimization methods require trajectory optimization to extract the dynamic parameters. The articles [14], [15], [18], and [25] do not have the same contributions as this article: The proposed methodology uses a CNN and returns the inertia, gravity, and friction dynamics parameters without a trajectory optimization.

The article is organized as follows. Section 2 describes the robotic arm and the control scheme. In section 3, the embedded system used to control the robot and data acquisition are shown, section 4 shows the parts of the parametric identification algorithm, section 5 shows the neural network training results, simulation results, and experimental results, and sections 6 and 7 shows the discussion and the conclusions.

II. ROBOT ARM

The system used for this investigation is a robot arm of 2 DoF that consists of two bars connected with two motors, as the schematic diagram of Fig. 1 shows. The dynamic model of the robot arm is shown in Eq. 1. The inertia matrix $M(\boldsymbol{q})$, the Coriolis matrix $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$, and the gravitational torque $\boldsymbol{g}(\boldsymbol{q})$ are directly determined by Euler-Lagrange equations, where $\boldsymbol{q} = [q_1, q_2]^T$ is the position of the robot, and $\dot{\boldsymbol{q}}$ is their derivate [12]. The viscous friction is modeled with a B matrix, and the Coulomb friction is modeled with a hyperbolic tangent with a constant λ and a K matrix [29].

$$\begin{aligned}
\boldsymbol{\tau} &= M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + B\dot{\boldsymbol{q}} + K\tanh(\lambda\boldsymbol{q}) \\
M(\boldsymbol{q}) &= \begin{bmatrix} I_1 + I_2\cos(q_2) & I_3 + 0.5I_2\cos(q_2) \\ I_3 + 0.5I_2\cos(q_2) & I_3 \end{bmatrix} \\
C(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \begin{bmatrix} -I_2\sin(q_2)\dot{q}_2 & -0.5I_2\sin(q_2)\dot{q}_2 \\ 0.5I_2\sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \\
B &= \mathrm{diag}(b_1, b_2),\ K = \mathrm{diag}(k_1, k_2) \\
\boldsymbol{q}(\boldsymbol{q}) &= [g_1\sin(q_1) + g_2\sin(q_1 + q_2),\ g_2\sin(q_1 + q_2)] \\
I_1 &= m_1 l_{c1}^2 + J_1 + m_2 l_1^2 + m_2 l_{c2}^2 + J_2 \\
I_2 &= 2m_2 l_1 l_{c2} \\
I_3 &= m_2 l_{c2}^2 + J_2 \\
g_1 &= m_1 l_{c1} g + m_2 l_1 g \\
g_2 &= m_2 l_{c2} g
\end{aligned} \tag{1}$$

TABLE 2. Dynamic parameters description.

	Parameters		Parameters	Units
m_1	Mass link 1	m_2	Mass link 2	kg
l_1	Length link 1	l_2	Length link 2	m
l_{c1}	Center of mass link 1	l_{c2}	Center of mass link 2	m
J_1	Inertia of link 1	J_2	Inertia of link 2	kg·m^2
g_1	Gravity torque 1	g_2	Gravity torque 2	kg·m^2/s^2
b_1	Viscosity joint 1	b_2	Viscosity joint 2	kg·m^2·s
k_1	Coulomb friction joint 1	k_2	Coulomb friction joint 2	kg·m^2/s^2
I_1	Composed inertia 1	I_2	Composed inertia 2	kg·m^2
I_3	Composed inertia 3			kg·m^2

A. CONTROL SCHEME

The proposed methodology requires that the robot follows a predefined trajectory to identify the parameters. The proportional derivative position control of Eq. 2 is designed for discrete-time [30]. The derivate of the position is estimated by subtracting the instant i minus the $i-1$, where i represents the time instant of sampling, and $\boldsymbol{q_{d_i}}$ is the desired position. Using the expansion of the Taylor series of the position $q_i = q_{i-1} + \frac{d}{dt}q_{i-1}h + \ldots + \frac{d^n}{dt^n}q_{i-1}h^n$ and taking only the two first terms of the right hand ($q_i \approx q_{i-1} + \frac{d}{dt}q_{i-1}h$), the velocity is estimated. The diagonal K_p matrix contains the proportional gains, and the diagonal matrix K_v has the derivative gains where the sampling time is implicit in its values.

$$\boldsymbol{\tau_{PDi}} = K_p(\boldsymbol{q_{d_i}} - \boldsymbol{q_i}) - K_v(\boldsymbol{q_i} - \boldsymbol{q_{i-1}}) \tag{2}$$

Because the proposed control scheme does not contain a measure of velocity, the stability demonstration is carried out

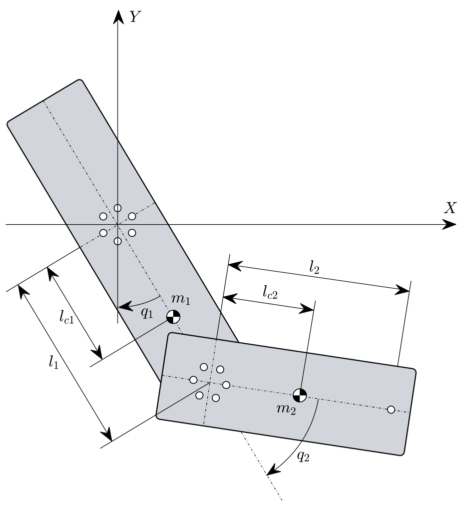


FIGURE 1. Schematic diagram of the robot arm.

with the input-to-state stability (ISS). The ISS establishes the bound of the state variables using the initial condition and the input: If the input is zero, the initial conditions bound the state variables: Eq. 1 must be local asymptotically stable in $\boldsymbol{q} = 0, \dot{\boldsymbol{q}} = 0$. Otherwise, both initial conditions and the input bound the state variables [31].

$$V(\boldsymbol{q}, \dot{q}) = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} + g_1[1 - \cos(q_1)] + g_2[1 - \cos(q_1 + q_2)] + \epsilon\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\boldsymbol{q} \quad (3)$$

The energy function of Eq. 3 demonstrates that Eq. 1 is asymptotically stable around $\boldsymbol{q} = \boldsymbol{0}, \dot{\boldsymbol{q}} = \boldsymbol{0}$ and $||\boldsymbol{q}|| < \pi$. The complete demonstration of the asymptotic stability of the robot is shown in appendix 1. The ISS analysis shows that the state variables (position and velocity) are bound with the initial condition vector and the upper bound of the input, as Eq. 4 shows:

$$||\boldsymbol{x}|| \leq W(||\boldsymbol{x}(t_0)||, t - t_0) + a_1 \left(\sup_{t_0 \leq \mu \leq t} ||\boldsymbol{\tau_{PD}}||(\boldsymbol{\mu}) \right) \quad (4)$$

where $\boldsymbol{x} = [\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T]^T$, t_0 is the initial time, a_1 is defined in appendix 1, $\frac{\partial}{\partial y_1}W(y_1, y_2) > 0$, and $\frac{\partial}{\partial y_2}W(y_1, y_2) < 0$.

III. EMBEDDED SYSTEM

The embedded system consists of interconnected hardware blocks into an FPGA to control the robot's position and send the articular position and torque by UART communication with a personal computer (PC). The embedded systems process both digital signals of the robot's encoders to read the position. A Pulse With Modulation (PWM) signal determines the value of torque for each motor: the torque is proportional to the duty cycle of the PWM signal. The desired trajectory of the robot is inside of internal RAM. Our hardware development of FPGA can execute the tasks of measuring the position, torque, and controlling the position, all done in 2.5ms. The hardware blocks are designed to handle appropriately 2 DoF, but the hardware design can take more than 2 DoF. Fig. 2 shows the developed hardware for the robot arm. First, the desired trajectory and control gains are received by serial communications from a PC. The hardware block (HB) **Write Ram** decodes the received data to program the **Ram-FPGA** HB where the single-precision float point is used. The **Write Ram** HB sends the bit start (*bst*) signal to initialize the robot arm control. The **Read Ram** HB puts in the *ctrlv* data bus the gains together with the pulse with modulation (PWM) frequency for each joint DC-motor. The *tryd* data bus contains the desired position for each time instant updated by the **Aq time** HB at 400 Hz.

The **Decoder** HB is used to determine the position of the robot arm and puts it in a 32-bit register. **PWM gen** generates the PWM signals: this block converts the register tau into two outputs pulsed signals. The **Fp ops** HB contains multiplication blocks, addition-subtraction, integer to float point, and float point to integer. The **PD** HB takes the control gains from the **Read Ram** HB together with the position of the robot. The error of position is calculated together with the velocity to determine the torque for the robot. The saturation function is implemented by checking the torque level, as the flow chart in Fig. 3 shows. The positions and torques are sent by serial communication to the PC. The **H bridge** generates a necessary clock signal to the H bridge of the robot's motors. The **Aq uart** HB makes a register of 64-bits that contains the position and torque of the robot. The first byte is the identifier, followed by two 16-bit registers used for each joint position. The torque uses an 11-bit register for both joints. The range for position register is -32768 to 32767 or 3.9 turns for each joint. The torque value ranges from -1024 to 1023: each unit represents 0.0978% of the duty cycle of PWM. The robot arm has two DC motors coupled with a 131.25 to 1 reduction gearbox. Each motor can develop up to 3 Nm with 12 volts. The resolution of the encoders is 0.748 $\times 10^{-3}$ rad. The maximum velocity of each joint is up to 8 rad/s and 2.52 m/s at the end of the second link. All the proposed embedded hardware design in Fig. 2 consumes 6683 of 22320 or 29.94% of the total logic elements of the FPGA. The inserted memory consumes 87% of the FPGA memory, and the clock frequency is 100 MHz. The hardware system proposal has been designed to fit the experimental board DE0-NANO© that includes the FPGA cyclon© IV EP4CE22F17C6 programmed with the Quartus© II software. This board contains enough logic elements to implement the control and acquisition system.

IV. IDENTIFICATION ALGORITHM

The proposed identification methodology is described in Fig. 4. First, the robot follows a predefined trajectory using a proportional-derivative (PD) control scheme: The required velocity for the PD is estimated using the sampled position points. The robot signals (torque, position, and estimations of velocity and acceleration) and an initial set of parameters $\boldsymbol{\beta}_n$ creates the input CNN image. The main idea of this image is when the introduced parameters are not coincident with

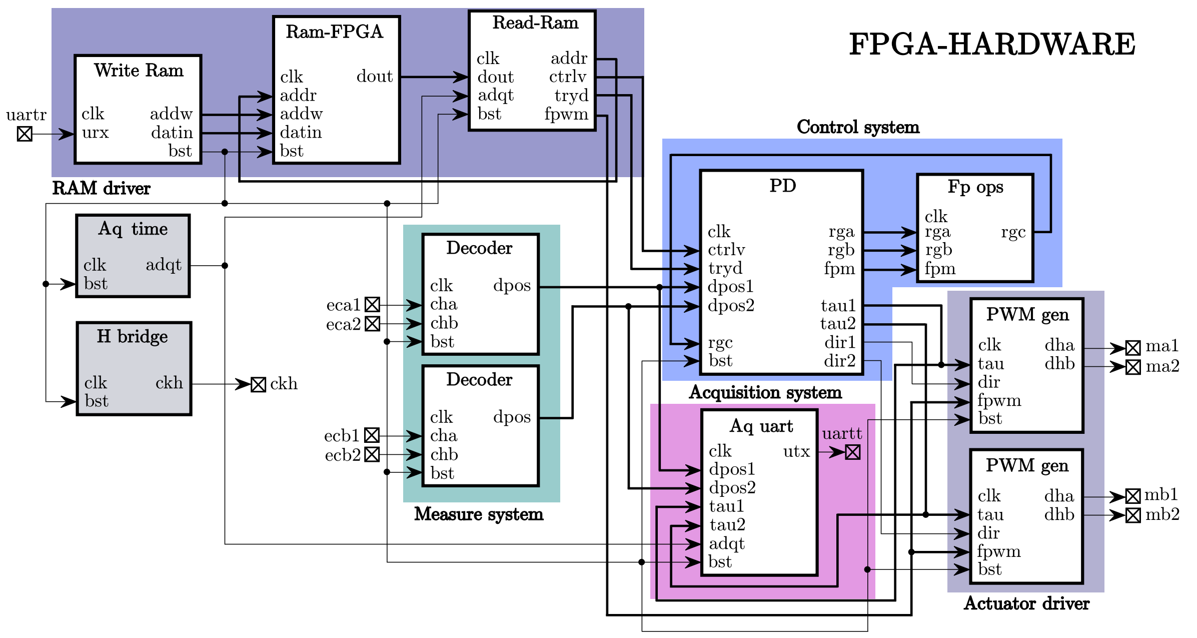


FIGURE 2. FPGA-based hardware design proposal.

the torque of the robot, the convolutional neural network returns the parametric residuals: i.e., the values that the initial set of parameters $\boldsymbol{\beta}_n$ needs to be the actual dynamic parameters of the robot. The residuals and the initial set of parameters $\boldsymbol{\beta}_n$ determine the estimated set of parameters $\boldsymbol{\beta}$ of the robot. The position, velocity, and acceleration signals, together with $\boldsymbol{\beta}$, reconstruct the torque $\boldsymbol{\tau}_r$ using the dynamic model of the robot. Comparing the similitude between the experimental torque $\boldsymbol{\tau}$ and its reconstruction $\boldsymbol{\tau}_r$ evaluates the identified parameters $\boldsymbol{\beta}$. Once the similitude percentage overcomes a predefined umbral u, $\boldsymbol{\beta}$ is adjusted to match the torque $\boldsymbol{\tau}$ values. The execution order of the proposed method is: 1) initialize $\boldsymbol{\beta}_n$, 2) Image creation, 3) CNN processing, 4) residual extraction, 5) obtain the estimated parameters $\boldsymbol{\beta}$, 6) torque reconstruction, 7) torque similitude evaluation, 8) similitude decision and 9) adjustment of $\boldsymbol{\beta}$. The details of the identification algorithm parts are found in subsections IV-A, IV-B, and IV-C.

A. TRAINING DATA GENERATION

The data creation for CNN training consists of the transpose multiplication of two vectors to obtain a matrix. In this case, this image has two channels because the robot has 2 DoF. Because the simulation of Eq. 1 takes time, and for each new set of parameters is necessary to adjust the control gains, a numeric substitution is used. A defined trajectory determines the velocity and acceleration to obtain the torque. The signals image of the robot needs to hold the most relevant information about the motion, and its size needs to be as small as possible. The 100×100×2 size was selected because it is small enough but maintains the 100 discrete cosine band frequency, i.e., 0 to 33 Hz at 2.5ms of the sampling period.

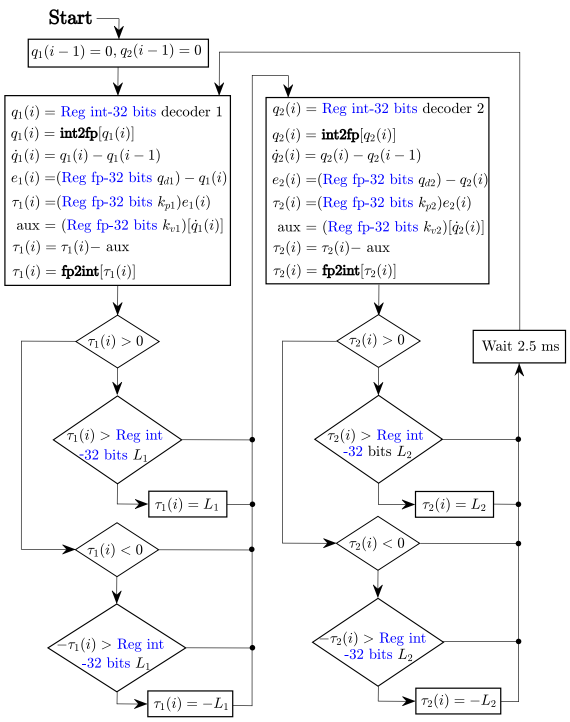
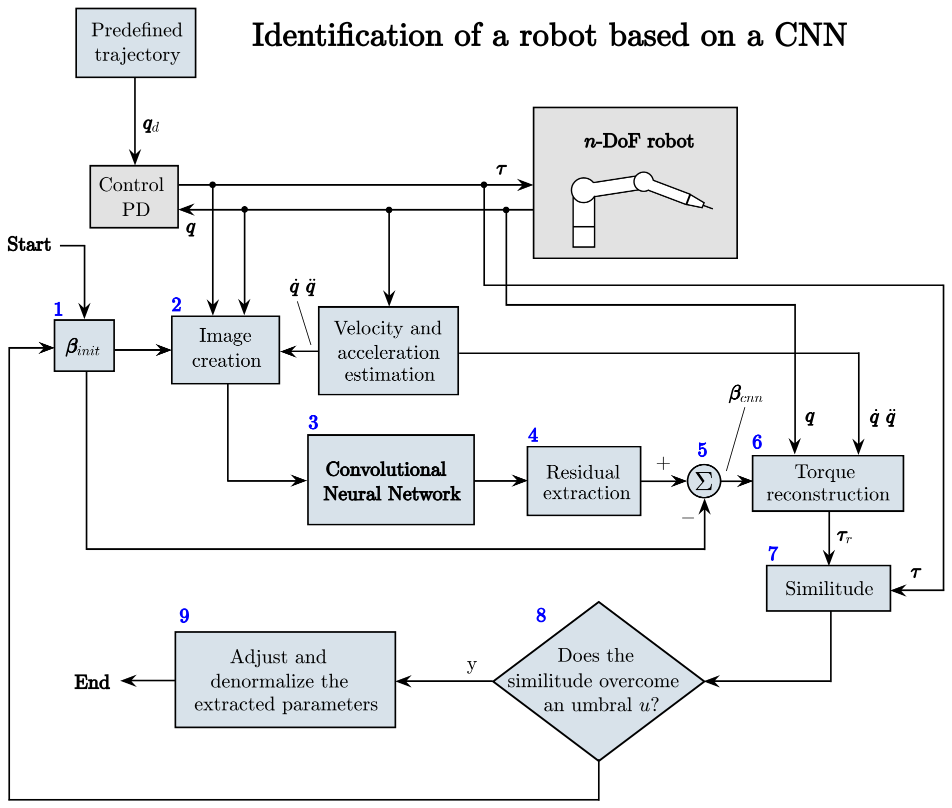

Start
$q_1(i-1)=0, q_2(i-1)=0$
$q_1(i)$ = Reg int-32 bits decoder 1
$q_1(i)$ = int2fp$[q_1(i)]$
$\dot{q}_1(i) = q_1(i) - q_1(i-1)$
$e_1(i)$ =(Reg fp-32 bits q_{d1}) $- q_1(i)$
$\tau_1(i)$ =(Reg fp-32 bits k_{p1})$e_1(i)$
aux = (Reg fp-32 bits k_{v1})$[\dot{q}_1(i)]$
$\tau_1(i) = \tau_1(i)-$ aux
$\tau_1(i)$ = fp2int$[\tau_1(i)]$
$q_2(i)$ = Reg int-32 bits decoder 2
$q_2(i)$ = int2fp$[q_2(i)]$
$\dot{q}_2(i) = q_2(i) - q_2(i-1)$
$e_2(i)$ =(Reg fp-32 bits q_{d2}) $- q_2(i)$
$\tau_2(i)$ =(Reg fp-32 bits k_{p2})$e_2(i)$
aux = (Reg fp-32 bits k_{v2})$[\dot{q}_2(i)]$
$\tau_2(i) = \tau_2(i)-$ aux
$\tau_2(i)$ = fp2int$[\tau_2(i)]$
$\tau_1(i) > 0$
$\tau_1(i) >$ Reg int -32 bits L_1
$\tau_1(i) = L_1$
$\tau_1(i) < 0$
$-\tau_1(i) >$ Reg int -32 bits L_1
$\tau_1(i) = -L_1$
$\tau_2(i) > 0$
$\tau_2(i) >$ Reg int -32 bits L_2
$\tau_2(i) = L_2$
$\tau_2(i) < 0$
$-\tau_2(i) >$ Reg int -32 bits L_2
$\tau_2(i) = -L_2$
Wait 2.5 ms

FIGURE 3. Flow chart of the PD control. L_1 and L_2 are equal to 1023.

The image contains enough information to extract the dynamic parameters with the CNN. In addition, the image is small for future embedded real-time parameter identification, where the image size determines the latency of identification. The image contains enough information to extract the dynamic parameters with the CNN. Consequently, the motion signals are subsampled with the technique visualized



Identification of a robot based on a CNN
Predefined trajectory
q_d
Control PD
q
τ
n-DoF robot
Start
End
1
β_{init}
2
Image creation
$\dot{q}\ \ddot{q}$
Velocity and acceleration estimation
3
Convolutional Neural Network
4
Residual extraction
5
+
−
β_{cnn}
6
Torque reconstruction
τ_r
7
Similitude
8
Does the similitude overcome an umbral u?
y
9
Adjust and denormalize the extracted parameters

FIGURE 4. General diagram of the proposed identification methodology.

in Fig. 5.

$$X(k) = \sqrt{\frac{2}{N}}\sum_{i=1}^{N} a(k)x(i)\cos[\frac{\pi}{2N}(2i-1)(k-1)]$$
$$a = [1+\delta(k-1)]^{-1/2}$$
$$\delta(z) = 1, \text{if and only if} z = 0 \qquad (5)$$

The discrete cosine transform version two (DCT-II) of Eq. 5 is used for signal subsampling, where X is the frequency spectrum, k is the frequency band, and δ is the Kronecker delta [32]. First, an input signal x of N samples is transformed with the DCT-II to obtain the frequency spectrum X. Second, X is cut to the desired integer length. N does not need to be a multiple of M. The third step is to apply the inverse DCT-II (iDCT-II) and adjust to the levels of the original signal x to obtain a subsampled signal x_s, as Fig. 5 shows. The identification algorithm uses estimations of velocity and acceleration based on the sampled position.

$$\dot{x} = \mathbf{dct2dev}(x, h, c_f) \qquad (6)$$
$$\dot{X}_z = \mathbf{DCT-II}(\dot{x}_z)$$
$$\dot{X} = \dot{X}_z \exp(-c_f\omega^2)$$
$$\dot{x} = \mathbf{iDCT-II}(\dot{X}) \qquad (7)$$

Eq. 7 shows the **dct2dev** function that has been designed to filter and derivate a sampled position. First, the derivate is

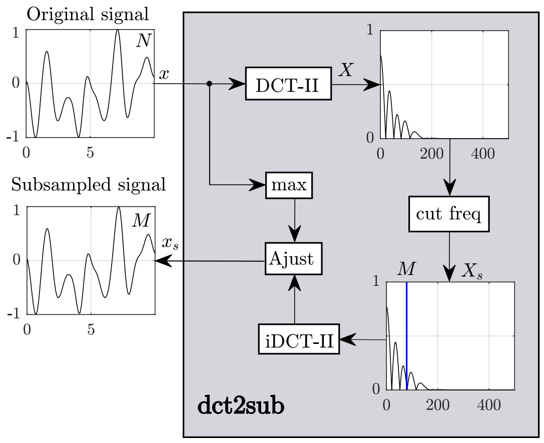


FIGURE 5. Subsampling technique based on the DCT-II.

estimated with $\dot{x}_z(i) = h^{-1}[x(i) - x(i-1)]$, where i is the time instant and h is the sampling time. Once the derivate is determined, a low-pass Gaussian filter of Eq. 7 multiplies the discrete cosine spectrum of the derivate to delete the noise, where ω is the discrete cosine spectrum frequency, and c_f is the filter coefficient. The robot signals and a set of parameters create the image using the subsampled and normalized signals of Eq. 8, where the subindex $()_s$ indicates that the signal is subsampled with the technique of Fig. 5.

$$
\begin{aligned}
\dot{\boldsymbol{q}} &= \mathbf{dct2dev}(\boldsymbol{q}, h, c_f)\\
\ddot{\boldsymbol{q}} &= \mathbf{dct2dev}(\dot{\boldsymbol{q}}, h, c_f)\\
\{\boldsymbol{\tau}_s, \boldsymbol{q}_s, \dot{\boldsymbol{q}}_s, \ddot{\boldsymbol{q}}_s\} &= \mathbf{dct2sub}(\boldsymbol{\tau}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, 100)\\
\boldsymbol{q}_r &= [q_{s1}/\max(q_{s1}), q_{s2}/\max(q_{s2})]^T\\
\dot{\boldsymbol{q}}_r &= [\dot{q}_{s1}/\max(\dot{q}_{s1}), \dot{q}_{s2}/\max(\dot{q}_{s2})]^T\\
\ddot{\boldsymbol{q}}_r &= [\ddot{q}_{s1}/\max(\ddot{q}_{s1}), \ddot{q}_{s2}/\max(\ddot{q}_{s2})]^T\\
\boldsymbol{\tau_r} &= \boldsymbol{\tau_s}/\max(\boldsymbol{\tau_s})
\end{aligned}
\tag{8}
$$

First, the vector $\boldsymbol{\alpha}$ of Eq. 9 contains the normalized and subsampled input torque $\boldsymbol{\tau_r}$. A $\boldsymbol{\psi}$ vector contains two times the torque $\boldsymbol{\tau_r}$ and the dynamic model $\boldsymbol{f}$ of Eq. 1 that reconstruct the torque using the input set of parameters $\boldsymbol{\beta}_n$: If $\boldsymbol{\beta}_n$ is the actual set of parameters, $\boldsymbol{\psi}$ is approximately equal to $\boldsymbol{\alpha}$. The first part of the image Z_1 is these two vectors multiplied in transpose form, as Eq. 9 shows.

$$
\begin{aligned}
\boldsymbol{\alpha} &= \boldsymbol{\tau}_r\\
\boldsymbol{\psi} &= 2\boldsymbol{\tau_r} - \boldsymbol{f}(\boldsymbol{q_r}, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r, \boldsymbol{\beta}_n)\\
Z_1 &= \boldsymbol{\psi}\boldsymbol{\alpha}^T\\
\boldsymbol{\alpha}_2 &= \mathbf{normalize}(\boldsymbol{\alpha}), \boldsymbol{\psi}_2 = \mathbf{normalize}(\boldsymbol{\psi})\\
\boldsymbol{\alpha_2} &= \log(\boldsymbol{\alpha_2} + 1 \times 10^{-3})\\
\boldsymbol{\psi_2} &= \log_{10}(\boldsymbol{\psi}_2 + 1 \times 10^{-3})\\
Z_2 &= \boldsymbol{\psi_2}\boldsymbol{\alpha}_2^T\\
Z &= [Z_1/\mathbf{max}(Z_1)] + [Z_2/\mathbf{max}(Z_2)]
\end{aligned}
\tag{9}
$$

The other image part Z_2 uses the normalized versions of $\boldsymbol{\alpha}$ and $\boldsymbol{\psi}$ called $\boldsymbol{\alpha_2}$ and $\boldsymbol{\psi_2}$: The **normalize** function range their argument from 0 to 1. The logarithmic functions log and $\log_{10}$ enhance some image regions with peaks in some parts: The convolutional neural network quickly identifies this image information due to their convolution operations. The image is finally constructed, adding parts Z_1 and Z_2, as Eq. 9 shows. The CNN works better with normalized input data [6]. Therefore, the signals of Eq. 8 are normalized. Observe that the torque cannot be normalized independently because Eq. 1 is coupled. Thus, the max value of the torques normalizes both signals.

The inertia parameters I_1, I_2, and I_3 and the gravitational parameters g_1 and g_2 are generated with the masses and lengths of the robot. Consequently, the parameters m_1, m_2, l_{c1}, l_{c2}, l_1, and l_2 are randomly generated for the data creation. For maintaining inertia and gravitational parameters around the unit, l_1, l_2, l_{c1}, l_{c2}, and J_1 are in the range of 1×10^{-3} to 0.1. For m_1, m_2 and J_2, the range is 1e-3 to 1. The friction coefficients range randomly from 1×10^{-3} to 1. The λ coefficient is set in 20 units for the Coulomb friction, and the predefined trajectory is shown in Eq. 10 and figure 6, where the time ranges from 0 to 3π.

$$
\begin{aligned}
q_1 &= \sin(t) + 0.1\cos(2.5t + 3)\\
q_2 &= -\cos(t) - 0.1\sin(2.5t + 3)
\end{aligned}
\tag{10}
$$

The trajectory of Eq. 10 does not lose its shape when the time is multiplied by a constant. The convolutional neural network extracts the parameters using only the shape of the trajectory and the torque. Therefore, all identified parameters can be adjusted to fit the time-scaled trajectory.

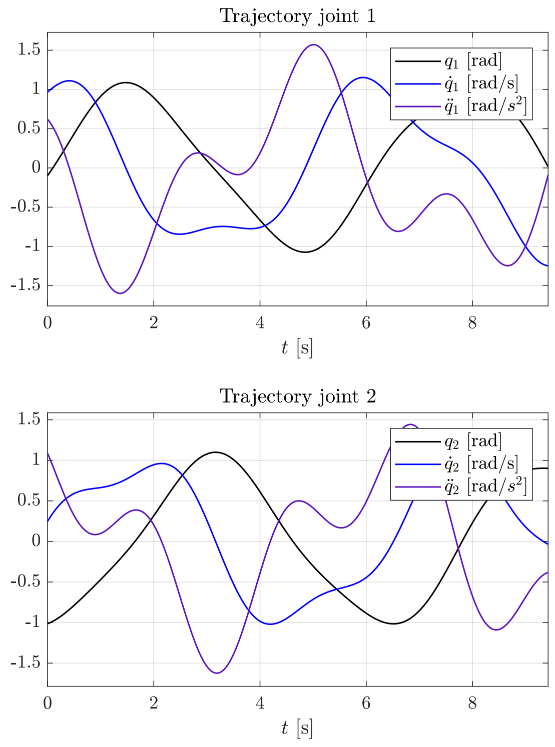


FIGURE 6. Predefined parameter identification trajectory.

The training set contains images M_p labeled with zero parameter residuals and M_n with non-zero parameter residuals. In the case of M_n, the torque signals and the motion signals correspond to the parameters. In the case of M_n, the signals do not match the parameters. M_p and M_n images are made with Eq. 9. The difference remains in the set of parameters introduced: M_n uses a set of parameters randomly selected in a range of 1×10^{-3} to 3.

B. CONVOLUTIONAL NEURAL NETWORK

The CNN extracts from the image of Eq. 9 the parameters residuals. If the creation of the image contains the set of parameters coincident with the torque in the function of the position, velocity, and acceleration, the residuals are zero. Because the output of the CNN can only range from 0 to 1 and the training data labels are the parameters residuals, the training process of the CNN evaluates the labels with the activation function of the output layer. The proposed CNN of Fig. 7 has an input of $100 \times 100 \times 2$ pixels, and the output are the nine parameters of the robot arm. Layers 1, 2, and 3 are convolutional, and 4, 5, and 6 are fully connected. The size of the kernels of the CNN is $9 \times 9 \times 2 \times 10$, $5 \times 5 \times 10 \times 10$, and $3 \times 3 \times 10 \times 10$ for the first, second, and third layers. For each convolutional layer, there is a bias vector of size 10×1 for each output map. In the first and second layers, there is a max pool layer of size 2×2. The CNN design of Fig. 7 is only

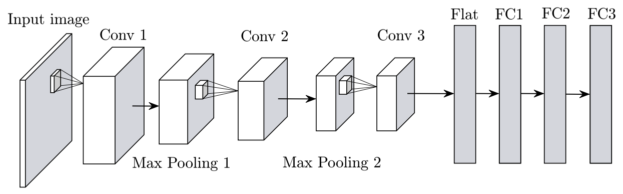


FIGURE 7. CNN architecture used.

TABLE 3. Convolutional neural network characteristics.

Layer	Size	Neurons	Synaptic weights	Activation function
Conv 1	92x92x10	-	1630	ReLU
Pool 1	46x46x10	21160	-	-
Conv 2	42x42x10	-	2510	ReLU
Pool 2	21x21x10	4410	-	-
Conv 3	19x19x10	3610	910	ReLU
FC 1	100x1	100	361100	ReLU
FC 2	100x1	100	10100	ReLU
FC 3	9x1	9	909	Sigmoid
Total	-	29389	377159	-

for the robot arm of 2 DoF: There is not a general design of the CNN for the proposed algorithm because it is not a trivial problem [6]. The CNN layers 1 and 2 contain reduction or Pool layers to concentrate the relevant extracted information in a small space. Pool layers are set with a window of 2×2 where the max value is extracted. All the convolutions are performed with Eq. 11, where $o = [0, O-1]$ is the output map index, Y_o is the output map, f_a is the activation function, $d = [0, D-1]$ is the input map index, X_d is the input map, b_o is the bias, and W_{do} is the kernel. The convolution of this equation is carried out without padding or striding [33].

$$Y_o = f_a\left[b_o + \sum_{d=0}^{D-1} \text{conv}(X_d, W_{do})\right] \tag{11}$$

The activation function of layers 1 to 5 is the Rectified Linear Unit (ReLU) of Eq. 13: The ReLU activation function prevents the vanishing of the error gradient in the hidden layers [6]. The output layers use the sigmoid function of Eq. 13 due to the smooth behavior where c is a constant to adjust the saturation level output layer. With this constant, it is possible to avoid the output layer taking only values of 0 or 1.

$$f(z) = \max(z, 0) \tag{12}$$
$$f(z) = [1+\exp(-cz)]^{-1} \tag{13}$$

The CNN features and size is shown in Table 3. The training of the CNN is implemented with the backpropagation error algorithm [33]. The learning rate is set as $\eta = 20\times10^{-3} + 0.2\times10^{-7}T$, where T is the training iteration. The training of the CNN is speeding with a constant $MT = 100$ used in the partial derivate of the loss function. The total training iterations n_T are set in 500 thousand, and the training data is split into two parts. Eight thousand images are used to train the CNN, and 2 thousand are for testing the training. The loss function is the mean square value of the output error $L = \frac{1}{18}\sum_{j=1}^{9}(y_{d-j} - y_{cnn-j})^2$, where $y_{d-j} = [1+\exp(L_{d-j})]^{-1}$, and L_{d-j} is the label of the training images.

C. IDENTIFICATION ALGORITHM PROPOSED

The proposed identification algorithm is displayed in **Algorithm 1**. The required inputs are the position and torque of the robot arm together with their dynamic model.

Algorithm 1 Parametric Identification of the Robot Arm Model.

```
Input: Position $\boldsymbol{q}$, torque $\boldsymbol{\tau}$, and dynamic model $\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\beta})$
    of the robot arm. Trained synaptic weights $\boldsymbol{W}, \boldsymbol{B}$ of the
    CNN
 1: $\dot{\boldsymbol{q}} \leftarrow$ dct2dev($\boldsymbol{q}, h, c_f$)
 2: $\ddot{\boldsymbol{q}} \leftarrow$ dct2dev($\dot{\boldsymbol{q}}, h, c_f$)
 3: $\boldsymbol{\tau} \leftarrow$ dct2low($\boldsymbol{\tau}, c_f$)
 4: $\{\boldsymbol{q_c}, \dot{\boldsymbol{q}}_c, \ddot{\boldsymbol{q}}_c, \boldsymbol{\tau_c}\} \leftarrow$ cut($\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau}, n_1, n_2$)
 5: $c_p \leftarrow$ fit2trt($\boldsymbol{\tau}_c$)
 6: $\boldsymbol{\tau}_c \leftarrow \boldsymbol{\tau}_c c_p$
 7: $\{\boldsymbol{q_s}, \dot{\boldsymbol{q}}_s, \ddot{\boldsymbol{q}}_s, \boldsymbol{\tau_s}\} \leftarrow$ dct2sub($\boldsymbol{q_c}, \dot{\boldsymbol{q}}_c, \ddot{\boldsymbol{q}}_c, \boldsymbol{\tau_c}, 100$)
 8: $u \leftarrow d_u$
 9: $i_1 \leftarrow N_{exe}$
10: $o_v \leftarrow L_{exe}$
11: $\boldsymbol{\beta}_m \leftarrow 0$
12: $\{\boldsymbol{q_n}, \dot{\boldsymbol{q}}_n, \ddot{\boldsymbol{q}}_n, \boldsymbol{\tau_n}\} \leftarrow$ bmax($\boldsymbol{q_c}, \dot{\boldsymbol{q}}_c, \ddot{\boldsymbol{q}}_c, \boldsymbol{\tau_c}$)
13: for $k_T = 1$ to $i_1$ do
14:     $n_e \leftarrow 0$
15:     $N_{it} \leftarrow 0$
16:     while $n_e < u$ and $N_{it} < o_v$ do
17:         $\boldsymbol{\beta}_n \leftarrow$ rand(9, range of $1\times10^{-3}$ to 3)
18:         $Z \leftarrow$ sig2img($\boldsymbol{q_n}, \dot{\boldsymbol{q}}_n, \ddot{\boldsymbol{q}}_n \boldsymbol{\tau_n}, \boldsymbol{\beta_n}$)
19:         $\boldsymbol{r} \leftarrow$ CNN($\boldsymbol{W}, \boldsymbol{B}, Z$)
20:         $\boldsymbol{r} \leftarrow -\log[(1-r)/r]$
21:         $\boldsymbol{\beta} \leftarrow \boldsymbol{r} + \boldsymbol{\beta_n}$
22:         if $\boldsymbol{\beta}$ is not physically possible then
23:             $\boldsymbol{\beta} \leftarrow \boldsymbol{0}$
24:             $n_e \leftarrow 0$
25:         else
26:             $\boldsymbol{\tau_r} \leftarrow \boldsymbol{f}(\boldsymbol{q_s}, \dot{\boldsymbol{q}}_s, \ddot{\boldsymbol{q}}_s, \boldsymbol{\beta})$
27:             $n_e \leftarrow 0.5$(metr2ev($\tau_{n1}, \tau_{r1}$)+
    metr2ev($\tau_{n2}, \tau_{r2}$))
28:         $N_{it} \leftarrow N_{it} + 1$
29:     $\boldsymbol{\beta_m} \leftarrow \boldsymbol{\beta_m}$ union $\boldsymbol{\beta}$
30: $\boldsymbol{\beta_o} \leftarrow$ high2sel($\boldsymbol{\beta_m}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau}$)
31: $\boldsymbol{\tau_r} \leftarrow \boldsymbol{f}(\boldsymbol{q_c}, \dot{\boldsymbol{q}}_c, \ddot{\boldsymbol{q}}_c, \boldsymbol{\beta_o})$
32: $\boldsymbol{\beta_o} \leftarrow \boldsymbol{\beta_o}[\max(\boldsymbol{\tau}_c)/(c_p \boldsymbol{\tau}_r)]$
Output: Dynamic parameters $\boldsymbol{\beta_o}$
```

Once the CNN has been trained, the algorithm needs the synaptic weights to return the residuals. First, the derivates of the position are estimated with the function **dct2dev** shown in Eq. 7. The sample time of the embedded system is $h = 2.5$ms, and the filter parameter c_f is set in 1×10^{-4}. Once the velocity and acceleration are estimated, the torque is filtered with the function **dct2low** with the same cut parameter c_f used for derivating as Eq. 14 shows.

$$\mathbf{dct2low}(x, c_f) = \mathbf{iDCT-II}[\mathbf{DCT-II}(x)\exp(-c_f\omega^2)] \tag{14}$$

The signals need to be cut to fit in the trajectory of Fig. 6 and Eq. 10. The function **cut** takes samples from the integer n_1 to n_2. The training data creates a mean torque $\boldsymbol{\tau_m}$ (see Fig. 9, distribution center torque) that adjusts the input torque $\boldsymbol{\tau}$ using the **fit2trt** function. Eq. 15 finds the constant c_p multiplied by the cut torque $\boldsymbol{\tau_c}$ where n_c is the length of $\boldsymbol{\tau_c}$, as lines 5 and 6 show.

$$c_p = \frac{1}{n_c}\sum_{k=0}^{n_c-1}\frac{||\boldsymbol{\tau_m}||}{||\boldsymbol{\tau_c}||} \tag{15}$$

In line 7, the signals are subsampled to 100 samples with the function **dct2sub** described in Fig. 5. In line 8, the variable u holds the umbral of similarity d_u: When the reconstructed torque and the input torque overcome this umbral, the algorithm considers that the parameters match the motion signals. Line 9 shows the variable i_1, which contains the number of executions of the identification process. The variable L_{exe} of line 10 is an internal limit for the while loop of line 16 to avoid that algorithm two going stuck. The execution results are stored in $\boldsymbol{\beta_m}$ initialized in zeros, as line 11 shows.

The set of signals is normalized by their maximum value, as described in Eq. 8 with the **bmax** function. The for loop of line 13 execute the identification process of the while of line 16. For each new while loop execution, the variable n_e where is stored the achieved similarity is set in zero. The variable N_{it} stores the CNN executions, and it is initialized in zero, as line 15 shows. The identification process of the while loop runs until the similarity umbral is overcome or N_{it} overcomes o_v. In line 17, an initial set of parameters $\boldsymbol{\beta_n}$ are initialized randomly in a range of 1×10^{-3} to 3. Then, the image Z is constructed with Eq. 9 with $\boldsymbol{\beta_n}$ and normalized by maximum value signals. This image is inserted in the CNN with the trained synaptic weights $\boldsymbol{W}$ and $\boldsymbol{B}$. The residual is recovered by applying the inverse sigmoid function, as line 20 shows. The actual set of parameters $\boldsymbol{\beta}$ is recovered, adding the residual $\boldsymbol{r}$ to the initial parameters $\boldsymbol{\beta_n}$. If the results are physically possible, the reconstructed torque of line 26 is made with the dynamic model and the subsampled signals. The torque similarity is put in the variable n_e. N_{it} increases one unit for each new iteration in line 28. In line 29, each parameter set $\boldsymbol{\beta}$ is stored in $\boldsymbol{\beta_m}$. The function **high2sel** of line 30 evaluates each set of parameters in $\boldsymbol{\beta_m}$ using the position, the estimations of velocity and accelerations, and the torque together with **metr2ev** to determine the best set of parameters that fit the real torque. The set makes the reconstructed torque of signals and the output parameter set $\boldsymbol{\beta_o}$. Finally, $\boldsymbol{\beta_o}$ is adjusted to fit the levels of the input torque, and it is returned.

The function **metr2ev** of **algorithm 2** evaluates the similarity of two signals in time and the DCT-II frequency spectrum. First, line 1 is calculated the similarity in time, and it holds in the variable a. Line 2 subtracts from the unit the value of a to return 100 percent when the signals are identical. In lines 3, 4, and 5, the frequency spectrum is calculated and normalized to determine the similarity in frequency in lines 6 and 7. Finally, both similarities are multiplied and adjusted with a power of 0.4 for balance between both similarities results.

Algorithm 2 Evaluation Metric **metr2ev**

```
Input: Signal 1 $x$, Signal 2 $y$
1: $a \leftarrow |\sum x - \sum y|[\sum(|x| + |y|)]^{-1}$
2: $a \leftarrow 1 - a$
3: $\{X, Y\} \leftarrow \mathbf{DCT-II}\{x, y\}$
4: $X \leftarrow X/\mathbf{max}(X)$
5: $Y \leftarrow Y/\mathbf{max}(Y)$
6: $b \leftarrow (\sum |X - Y|)(\sum |X| + |Y|)^{-1}$
7: $b \leftarrow 1 - b$
8: $z \leftarrow (ab)^{0.4}$
Output: Similarity $z$
```

V. RESULTS

This section describes the results of the training of the CNN with the images created with the proposed transformation technique, the numeric results, and the experimental results of the identification of a robot arm. For the training results of the CNN, the behavior of the loss functions indicates that the synaptic weights are adequately adjusted to meet the goal of the CNN. The numeric results show that the algorithm identifies the parameters, and the torque signal corresponds to the original in the validation. The experimental results demonstrate that the algorithm can determine the parameters of a real arm robot of 2 DoF. The identified parameters are validated with four trajectories.

A. TRAINING OF THE CNN

The images shown in Fig. 8 are taken from the data set for CNN training. The M_p images show the shapes formed when the parameters, the signals, and torques match. In the M_n images, the parameters used for their construction do not match the torque signals. The color representation is carried out with the color palette (R, G, B) of Eq. 16, where Z_{DoF1} and Z_{DoF2} are the channel 1 and 2 of the image of Eq. 9, respectively.

$$\begin{aligned} R &= (2/3)Z_{DoF1} + (0.7/3)Z_{DoF2} \\ G &= (2/3)Z_{DoF1} \\ B &= (2/3)Z_{DoF1} + Z_{DoF2} \end{aligned} \tag{16}$$

The 10,000 sets of training parameters create a torque distribution shown in Fig. 9. These figures show that the torque signals can be diverse under the predefined trajectory. The loss function evolution is displayed in figure 10. For the

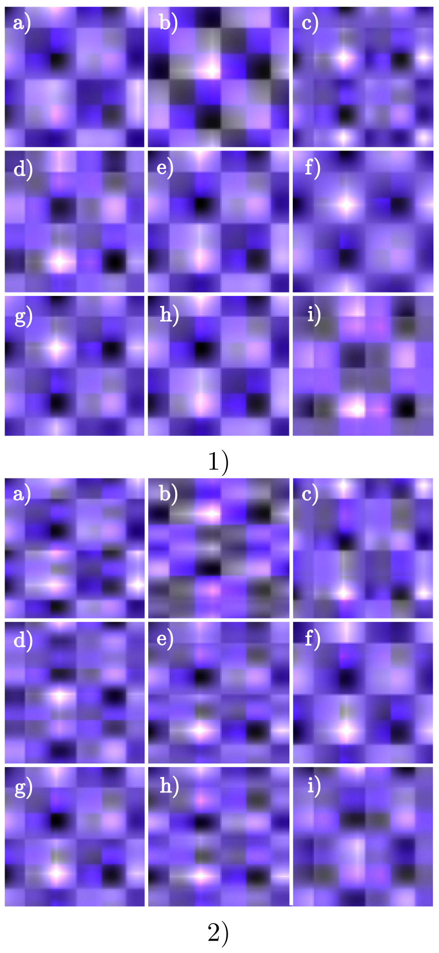


FIGURE 8. Images from the training data: 1) M_p images, 2) M_n images. Each image (from a) to i)) of M_p and M_n has the same torque and position signal.

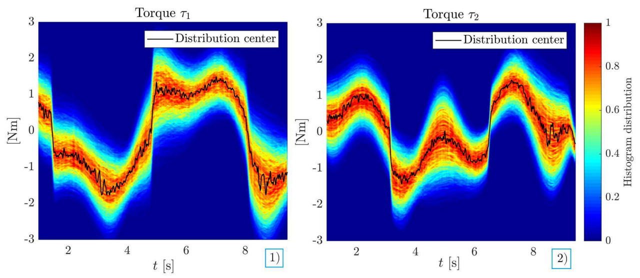


FIGURE 9. Torque distribution of the training data: 1) torque joint 1, 2) torque joint 2.

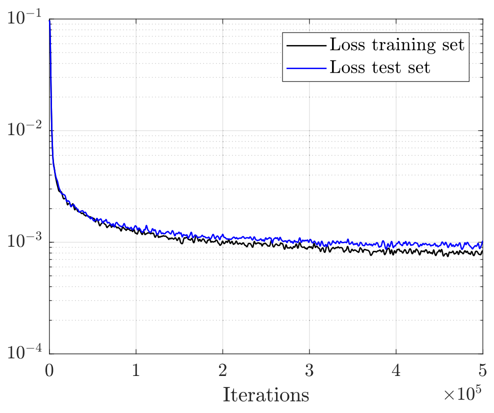


FIGURE 10. Loss function of the CNN training process.

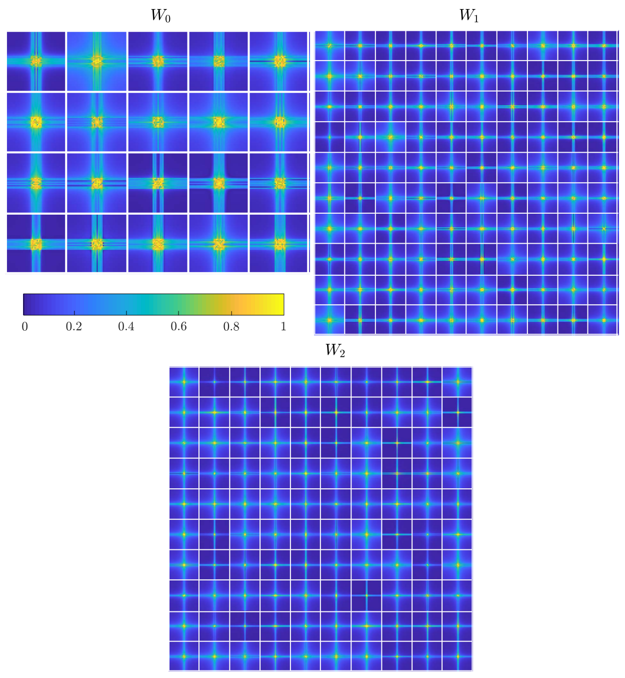


FIGURE 11. Normalized power spectrum of the convolution kernels.

training set, the square value of the error tends to 1e-3. In the case of the test set, it is observed that it is slightly over the train set. The CNN has been trained half-million times with 80% of images for training and 20% for testing with the gradient descent. The spectrum of the kernels of the convolutional layers is shown in Fig. 11. The shapes of the kernels are different and, in combination, can return the residuals of the convolutional layer. For the visualization of the kernels, the normalized Fourier transform is used in Fig. 11.

B. NUMERIC RESULTS

The algorithm is proved with four sets of parameters with their torques determined by direct substitution of the position, velocity, and acceleration of Eq. 1. The selected parameters are not part of the training data. Table 4 and Table 5 show the original parameters and the identified parameters. Table 6 shows the similarity evaluation with **algorithm 2**. The signals of the numerically determined torque and the reconstructed torque are similar, as shown the Fig. 12 and 13, where the subindex *cnn* indicates the reconstructed torque with the output parameters $\boldsymbol{\beta_o} = [I_1, I_2, I_3, g_1, g_2, b_1, b_2, k_1, k_2]^T$ of **algorithm 1**.

The evaluation of similarity with the proposed metric **metr2ev** shows that the numeric torque constructed with the set of parameters (N1, N2, N3, and N4) and the reconstructed torque with the extracted parameters with the algorithm

TABLE 4. Numeric results of identification, part 1 of 2. N1 and N1 are the numeric test parameters 1 and 2: N1 cnn and N2 cnn are the identified parameters of numeric test 1 and 2 with **algorithm 1**.

Parameters	N1	N1 cnn	N2	N2 cnn
I_1 kg·m^2	0.6590	0.7389	0.6259	0.4084
I_2 kg·m^2	0.4156	0.2872	0.4901	0.5188
I_3 kg·m^2	0.5578	0.5274	0.8725	0.8712
g_1 kg·m^2/s^2	0.9296	0.9710	0.5148	0.2812
g_2 kg·m^2/s^2	0.1349	0.0997	0.7013	0.6953
b_1 kg·m^2·s	0.3850	0.3749	0.3372	0.3556
b_2 kg·m^2·s	0.4875	0.4080	0.0574	0.0066
k_1 kg·m^2/s^2	0.1472	0.2244	0.0345	0.0317
k_2 kg·m^2/s^2	0.1044	0.1135	0.1668	0.2020

TABLE 5. Numeric results of identification, part 2 of 2. N3 and N4 are the numeric test parameters 3 and 4: N3 cnn and N4 cnn are the identified parameters of numeric test 3 and 4 with **algorithm 1**.

Parameters	N3	N3 cnn	N4	N4 cnn
I_1 kg·m^2	0.5020	0.4618	0.6944	0.5657
I_2 kg·m^2	0.9779	1.0288	0.5271	0.6752
I_3 kg·m^2	0.5040	0.4186	0.3408	0.2604
g_1 kg·m^2/s^2	0.4777	0.5445	0.3536	0.4467
g_2 kg·m^2/s^2	0.5724	0.4643	0.8523	0.7254
b_1 kg·m^2·s	0.0715	0.1200	0.5325	0.5101
b_2 kg·m^2·s	0.3820	0.5042	0.4714	0.5784
k_1 kg·m^2/s^2	0.0972	0.0474	0.1361	0.0920
k_2 kg·m^2/s^2	0.5721	0.5091	0.2451	0.1802

TABLE 6. Similarity numeric results. N1, N2, N3, and N4 indicates the similarity results for numeric test 1, 2, 3, and 4.

metr2ev	N1	N2	N3	N4
Time τ_1	99.94%	99.98%	99.99%	99.74%
Freq. τ_1	97.44%	97.76%	98.34%	97.73%
Total τ_1	**97.38%**	**97.74%**	**98.33%**	**97.48%**
Time τ_2	99.91%	99.84%	99.76%	99.67%
Freq. τ_2	98.29%	98.95%	98.59%	98.15%
Total τ_2	**98.20%**	**98.80%**	**98.35%**	**97.83%**

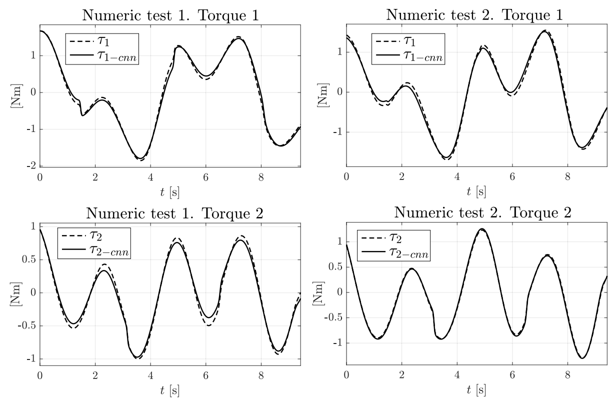


FIGURE 12. Numeric torques and reconstructed torques, test 1 and 2.

(N1 CNN, N2 cnn, N3 cnn, N4 cnn) are too close in time and frequency. However, the time part of **metr2ev** tends to be 100% when in reality, the torques of Fig. 12 and 13 are not identical. The frequency part reveals differences between the DCT spectrums of the numeric torques and the reconstructed torques, as Fig. 14 shows. The parameters identified by the

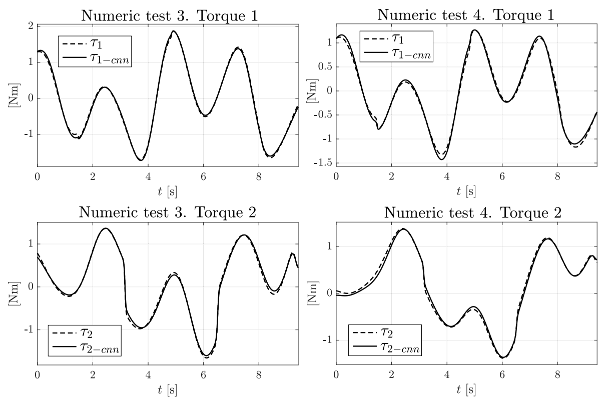


FIGURE 13. Numeric torques and reconstructed torques, test 3 and 4.

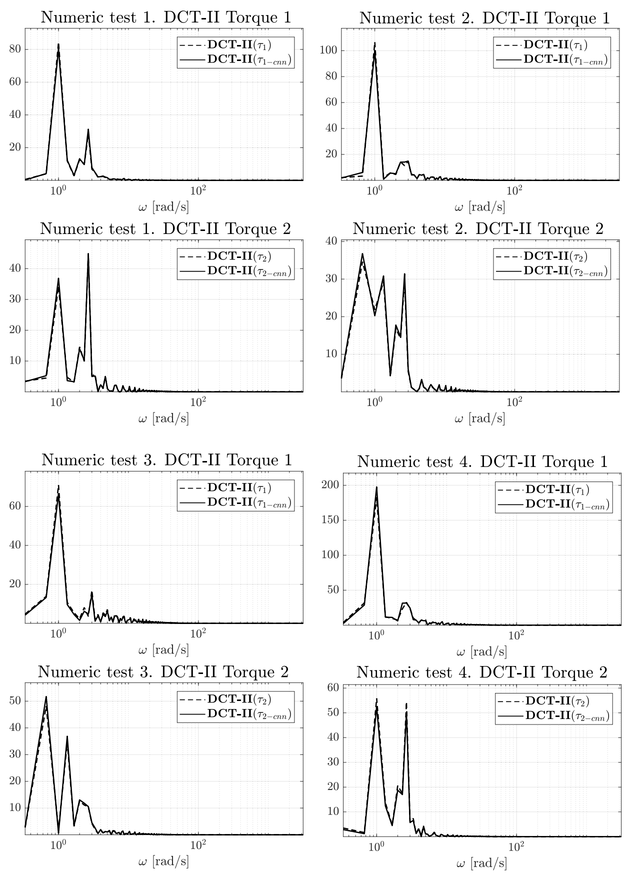


FIGURE 14. DCT of numeric torques and reconstructed torques.

CNN (N1 cnn, N2 cnn, N3 cnn, N4 cnn) are validated with a random trajectory constructed with a vector of random numbers in a range of 0 to 1. Then this vector is filtered with Eq. 14.

The numeric validation results show the reconstruction of the torque with another trajectory. Fig. 15 shows the validation torques in the function of the random trajectory of Fig. 16. Notice that even with another different trajectory of the identification trajectory of equation 10, the identified parameters by **algorithm 1** can reconstruct the torque signal.

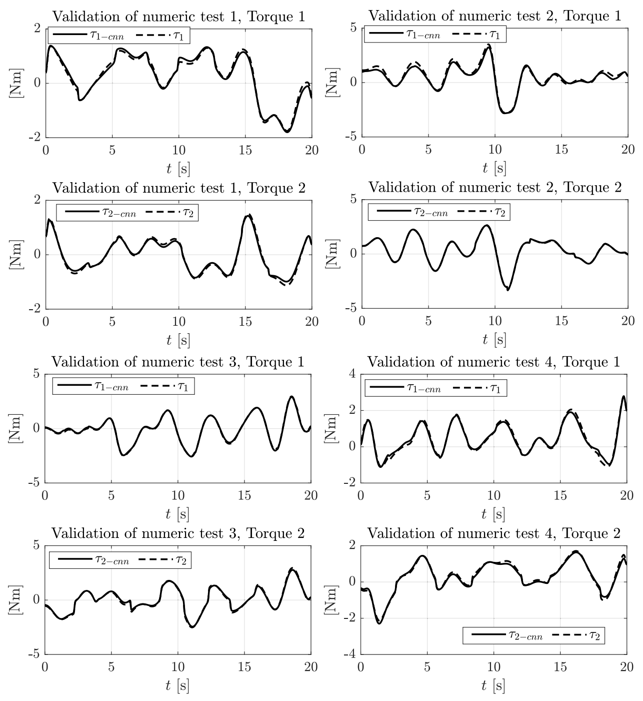


FIGURE 15. Torque validation of numeric tests.

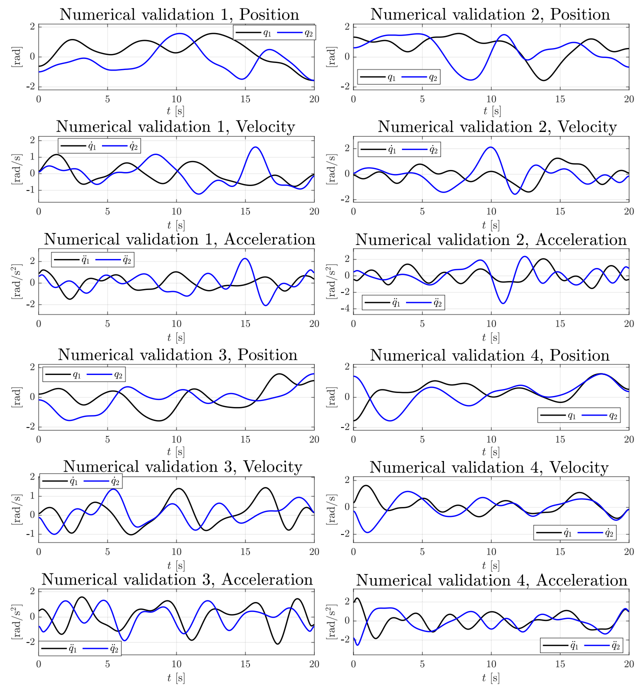


FIGURE 16. Trajectory validation of numeric tests.

The similarity values of Table 7 show that there are tiny differences between the DCT spectrum of Fig. 17. There are some differences in the actual and the identified parameters, as Tables 4 and 5 show. However, it is impossible to measure the parametric error because, in the real identification, the torque and position are the unique direct observable signals.

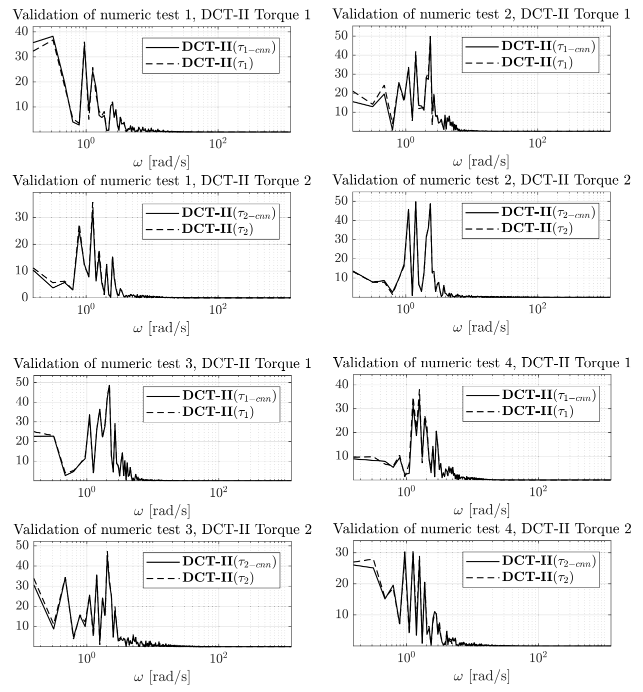


FIGURE 17. DCT of numeric validation torques.

TABLE 7. Similarity validation numeric results: N1, N2, N3, and N4 indicates the validation of test 1, 2, 3, and 4, respectively.

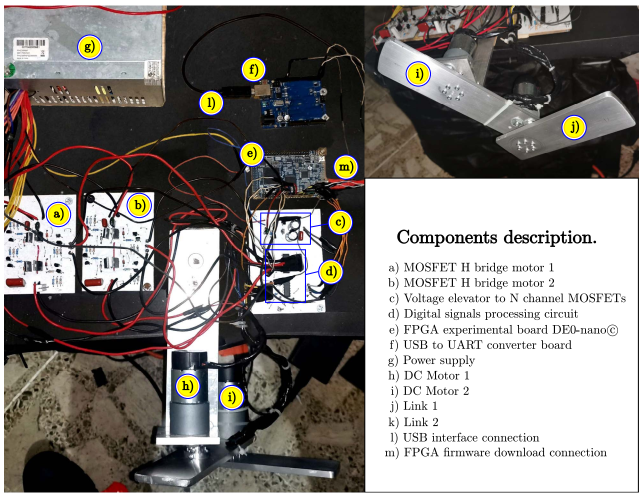
metr2ev	N1	N2	N3	N4
Time τ_1	99.77%	97.32%	99.61%	99.85%
Freq. τ_1	97.32%	97.59%	98.67%	97.36%
Total τ_1	**97.10%**	**94.98%**	**98.28%**	**97.22%**
Time τ_2	99.68%	99.94%	99.61%	99.01%
Freq. τ_2	98.59%	99.10%	98.62%	97.46%
Total τ_2	**98.25%**	**99.04%**	**98.24%**	**96.49%**

The metric **metr2ev** returns these differences in values that are not too close to 100%.

C. EXPERIMENTAL RESULTS

Fig. 18 shows the electronic components of the robot arm. The embedded hardware programmed in the experimental board DE0-nano˙ establishes serial communication with a USB to UART conversion board to send the robot signals to a PC. The MOSFET H bridges require 24 volts to operate correctly with a power voltage of 12 volts. The experimental trajectory made for identifying the parameters of the robot arm in Fig. 18 is displayed in Fig. 19. As it is visualized, the experimental position, velocity, and acceleration follow the predefined trajectory of Eq. 10. The proposed **algorithm 1** identifies the dynamic parameters of a real robot arm of two degrees of freedom. The LS method identifies the dynamic parameters of the trajectory in Fig. 19 to compare with **algorithm 1**.

Table 8 shows the similarity between the desired position and the experimental position of the robot arm. The gains of the control scheme K_p Nm/rad, and K_v Nm/rad, the PWM frequency of the motor, and the filter coefficients: The gains



Components description.

a) MOSFET H bridge motor 1
b) MOSFET H bridge motor 2
c) Voltage elevator to N channel MOSFETs
d) Digital signals processing circuit
e) FPGA experimental board DE0-nano©
f) USB to UART converter board
g) Power supply
h) DC Motor 1
i) DC Motor 2
j) Link 1
k) Link 2
l) USB interface connection
m) FPGA firmware download connection

FIGURE 18. Experimental setup and their components description.

TABLE 8. Constants of the embedded system with the control performance.

Joint	**metr2ev**	k_p	k_v	f_{PWM}	c_f
q_1	99.67%	180.4817	270.7266	1 kHz	1×10^{-4}
q_2	99.02%	30.6819	102.2730	1 kHz	1×10^{-4}

TABLE 9. Identified parameters of the robot arm of Fig. 18.

Parameter	**Algorithm 1**	LS
I_1 kg·m^2	0.0963	0.0076
I_2 kg·m^2	0.0319	0.0710
I_3 kg·m^2	0.0254	0.0096
g_1 kg·m^2/s^2	0.7876	0.6566
g_2 kg·m^2/s^2	0.08	0.1384
b_1 kg·m^2·s	0.0062	0.1827
b_2 kg·m^2·s	0.1943	0.3338
k_1 kg·m^2/s^2	0.3395	0.1538
k_2 kg·m^2/s^2	0.2585	0.2762

of the PD control have been chosen arbitrarily to have a response in the predefined identification trajectory to be used by the CNN proposed. The obtaining torque is determined by multiplying the PWM duty cycle by 2.7192/1023 and 2.7314/1023 for joints 1 and 2, respectively.

Table 9 shows the identified parameters using the **algorithm 1** and LS. Notice that g_1 has larger values than the inertia parameters for both identification methods: The robot arm concentrates the mass in the first articulation. The viscous friction b_1 and b_2 have low values than the Coulomb friction using the **algorithm 1**, while LS returns higher viscosity friction values.

Fig. 19 shows the experimental torque of both joints and their reconstructions using Eq. 1 and the identified parameters of Table 9. Notice that the torque reconstruction with the **algorithm 1** (blue line) and LS (red line) is close to the experimental signals. Fig. 20 shows that the DCT spectrum of the torque signals of Fig. 19 is close to the experimental torque.

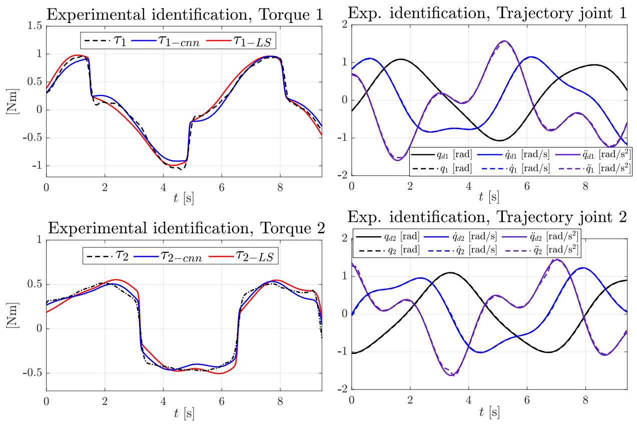


FIGURE 19. Experimental torque with the reconstructed torque, and trajectory of the robot of Fig. 18.

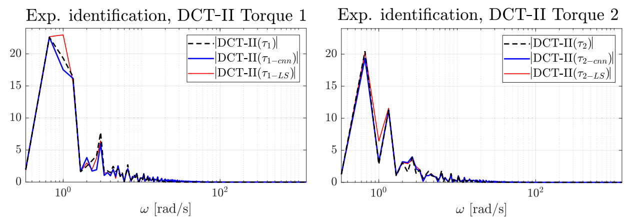


FIGURE 20. Experimental DCT spectrum of the real and reconstructed torque of the robot of Fig. 18.

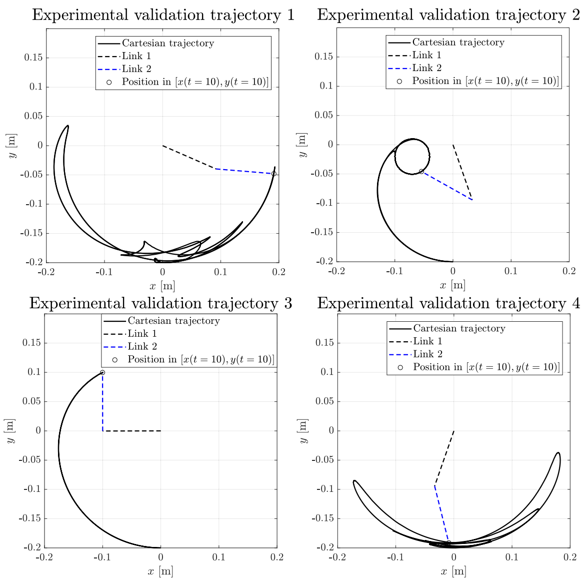


FIGURE 21. Cartesian trajectories of the validations tests.

The validation trajectories are shown in Fig. 21 and Fig. 22, where the PD scheme controls the robot to follow these validation trajectories. The first validation test is a filtered random signal, the second is a cartesian circle, the third is a profile, and the four is a sinusoidal trajectory, as Fig. 22 shows.

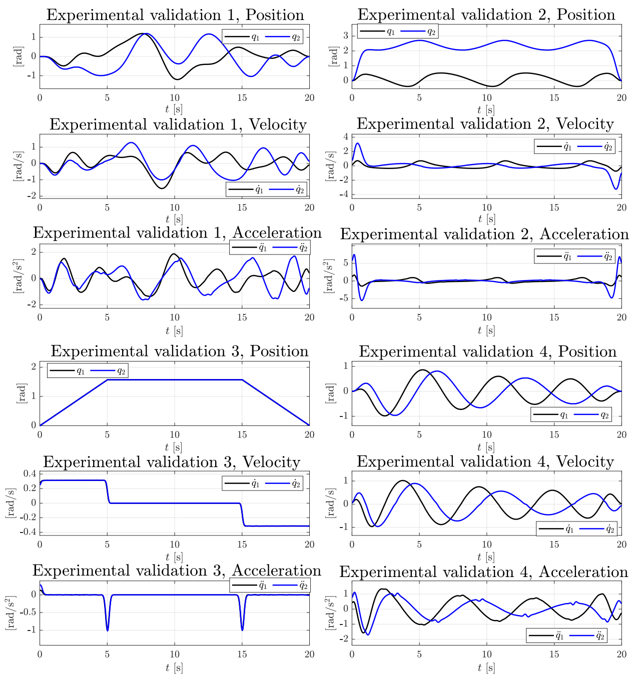


FIGURE 22. Experimental validation trajectory.

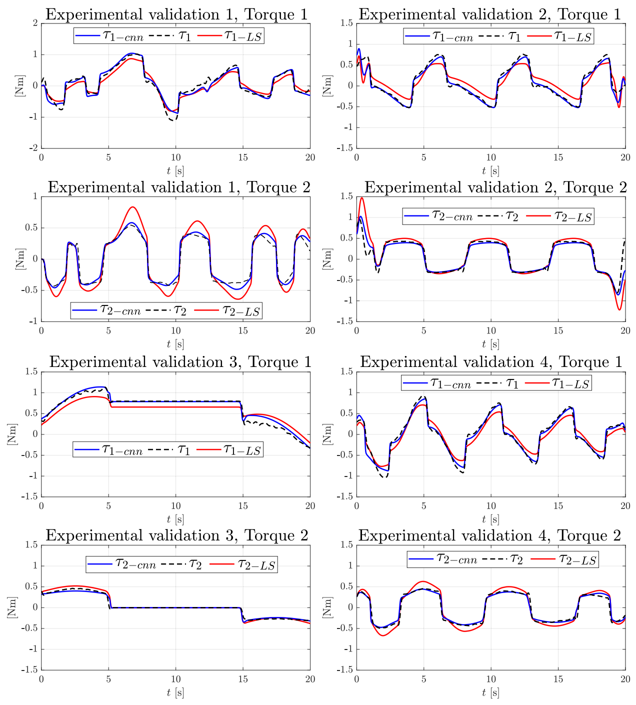


FIGURE 23. Experimental torque validation with the **algorithm 1** and LS.

TABLE 10. Similarity experimental results.

metr2ev Algorithm 1	ID	T1	T2	T3	T4
Time τ_1	99.16%	99.85%	98.64%	99.05%	99.91%
Freq. τ_1	94.34%	91.85%	91.25%	94.81%	93.21%
Total τ_1	**93.55%**	**91.72%**	**90.01%**	**93.91%**	**93.12%**
Time τ_2	99.91%	99.59%	99.11%	99.55%	99.51%
Freq. τ_2	94.60%	92.28%	91.01%	92.58%	91.56%
Total τ_2	**94.51%**	**91.90%**	**90.02%**	**92.16%**	**91.11%**
metr2ev LS	ID	T1	T2	T3	T4
Time τ_1	99.53%	97.95%	97.67%	98.19%	99.64%
Freq. τ_1	94.72%	91.31%	85.22%	92.39%	91.73%
Total τ_1	**94.27%**	**89.44%**	**83.23%**	**90.71%**	**91.41%**
Time τ_2	99.80%	99.98%	98.65%	98.17%	99.80%
Freq. τ_2	94.04%	91.67%	89.69%	91.98%	90.96%
Total τ_2	**93.86%**	**91.65%**	**85.82%**	**90.30%**	**90.78%**

The experimental validation torque signals and their reconstructions are in Fig. 23. Observe that even when the LS torque reconstruction is close to the experimental torque in Fig. 19, the validation results show that the LS parameters fail to reconstruct the validation torques. Fig. 24 shows several differences between reconstruction torque with LS in the frequency domain. Table 10 shows the similarity values of the experimental torque and its reconstruction using LS and the **algorithm 1**. For the identification trajectory (ID), LS has a better response in the joint one torque. However, for the validation test (T1, T2, T3, and T4), the parameters extracted with the **algorithm 1** overcome the LS results. The proposed evaluation metric **metr2ev** shows that the time similarity shows high values in Table 10. However, the frequency similarity determines that there are variations, as Fig. 24 shows.

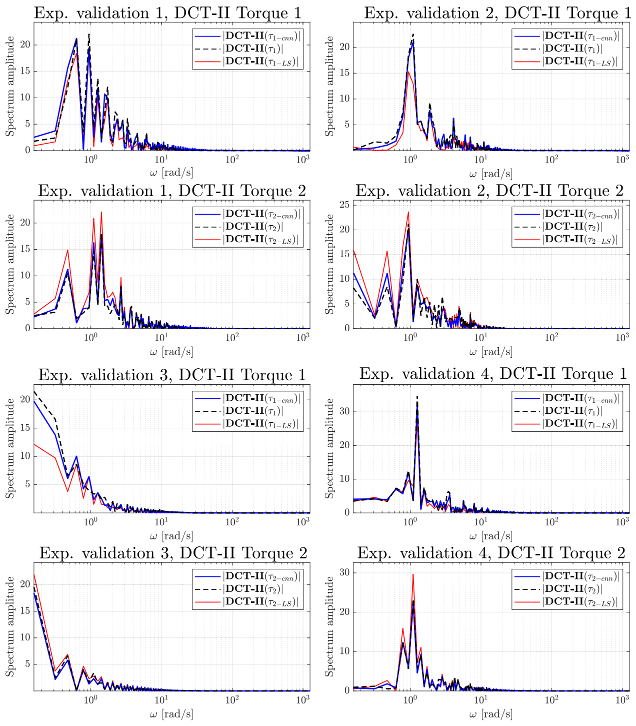


FIGURE 24. Experimental DCT spectrum of the validation torques.

VI. DISCUSSION

The proposed methodology identifies the parameters of a robot arm with a predefined trajectory. The training data are created by the torque signals obtained by numeric substitution of the position, velocity, acceleration, and random parameters. Instead of creating 10×10^3 simulations of Eq. 1, the torque creation with this technique reduces the time of the data creation significantly. With the numeric substitution, the tunning of the gains of the proposed control scheme is avoided. The only gains that need to be tuned come from the real robot. Notice that one of the requirements for the proposed methodology is that the robot arm follows the predefined trajectory. However, it is not critical that the trajectory performed by the robot be identical to the desired trajectory due to the CNN generalization feature: Inputs data that are similar in shape to the training data can return residuals parameters.

The functions of estimating derivates, filtering, and subsampling are based on the DCT-II because they placed the relevant spectrum information in the first frequency samples. The derivating of a quantized signal as the encoder's positions creates noise due to the loss of amplitude information. There is not possible to measure the actual velocity and acceleration with an encoder. Only an estimation can be done [10]. Filtering the estimate derivates with Eq. 7 returns a relative measure of velocity and acceleration. The subsampling technique achieves the reduction of a signal to any integer length. The signal to be subsampled with the technique shown in Fig. 5 needs to keep most of the spectrum information up to $(M-1)/2hN > f_x$, where f_x is the signal frequency spectrum. Otherwise, the signal must be filtered before subsampling. In the proposed **algorithm** 1, the position, velocity, acceleration, and torque are filtered to ensure the subsampling condition.

The proposed image creation with motion signals allows it to be processed with a CNN. The CNN can analyze images by regions in the convolutional layers. The features extracted in the convolutions allow 100 by 100 pixels where the essential information is condensed. Training data sets are constructed with a set of motion signals, a parameter set where these signals match (M_p image construction), and a random parameter set unrelated to motion signals (M_n image construction). Observe that the parameter set for M_n images has a greater range than that for Mp images. The idea behind this is that high parameter values can distort the torque made with the dynamic model. As a consequence, the CNN can identify the residuals of parameters quickly.

The proposed evaluation metric **metr2ev** implements the similitude analysis. The actual and reconstructed torques are evaluated in the time and frequency domain. The main reason is that the motion signals contain information in their spectrum that can be difficult to evaluate with only the time domain. The spectrums of the reconstructed torque signals in Fig. 20 show that they are close to the actual torque. However, the **metr2ev** returns high levels of similitude in time, but the same does not occur in frequency. The proposed metric reflects that the signals are close, but there are slight deformations that the time part of the **metr2ev** can not visualize, as shown in Table 9 and Fig. 19. In the case of the **metr2ev** applied to evaluate the control performance, it was observed that the position signal of the robot arm is too close to the desired input in the time and frequency domain, as shown in Table 8. The numeric results show that the algorithm can identify the parameters to fit a reconstructed torque, as displayed in Table 4 and Fig. 12. In this case, the model is the same for training data creation and does not contain noise by quantization or another type of noise. Notice that the reconstructed torque overcomes 97% of similitude even when the parameters are not identical.

The selected umbral for numeric tests is set in 92.82% with $N_{exe} = 100$ number of executions. The umbral is set in 92.37% for experimental data with $N_{exe} = 100$ number of executions. These constants are selected by experimentation; a high umbral of evaluation metric slows the algorithm and several executions close to 1 return a lower similitude level than if $N_{exe} > 1$. The LS parametric identification shows that the first joint torque similarity value overcomes the **algorithm 1**, as Table 10 shows. However, in the validation trajectories, the parameters of LS can not reconstruct the torque signals as the parameters of **algorithm 1** do. The torque signal contains deformations reflected in the evaluation metric due to the unmodeled components of the DC motors. However, the results show that a reconstructed torque overcomes 93% of similitude using the **algorithm 1**. Test 1 is obtained by a filtered uniform random distribution vector: $\boldsymbol{q_d}$ =**dct2low**[**rand**N, range -1.2 to 1.2]. Test 2 is a circle of 3.15 cm, test 3 is a motion profile, and test 4 is a sinusoidal trajectory. The selected test trajectories show that the model can reconstruct the torque for different motion signals if and only if the velocity of the joints does not exceed their maximum velocity value and the output torque does not overcome the max torque level. The measured torque and reconstructed torque behavior are similar, even when the robot arm holds a stationary position with the **algorithm 1** parameters. Tests 1, 2, and 4 have a Gaussian function $f_g = 1 - e^{-0.2(t)^2} - e^{-0.2(\max t - t)^2}$ multiplied by the test trajectory to prevent position, velocity, and acceleration overshoots.

VII. CONCLUSION

In this article, the parameter identification of a robot arm of 2 DoF with a methodology based on a CNN was described. The input data was constructed with a signal to image transformation technique that condenses the relevant motion information into a small space. The main idea of the algorithm consisted remains in the residual parameter extraction: The CNN returns how much the initial parameters are far away from the actual parameters of the robot to be identified. The training data has been created by numeric substation instead of numeric simulation of robot arm model. This technique makes the algorithm attractive because it only takes time in the CNN training but not data generation. Therefore, the results of numeric torque and the experimental data of the CNN training were validated. The discrete cosine transform has been used because of its property to compress the information in a few frequency bins: This represents an advantage in subsampling because it is possible to cut a signal into any integer length if and only if the frequency spectrum matches the condition of Nyquist. The control of the robot arm and data acquisition was implemented in real-time with the embedded system based on an FPGA. Instead of a sequence of execution of these tasks, the FPGA executes both simultaneously, each sample time of 2.5ms. The system in Fig. 2 can be scaled up to satisfy the DoF of a determined robot; in this case, the system handles two joints. The experimental validation results reveal that the model with the identified parameters set can reconstruct the torque with up to 93.91% similarity using the proposed algorithm. The parametric identification with LS can not reconstruct the validation torques as the proposed methodology. These results show that the proposed methodology can be helpful for robotic applications that need the parameters without an optimization trajectory step.

APPENDIX A: INPUT TO STATE STABILITY

The ISS requires that Eq. 1 is asymptotically stable around the origin. The energy function of Eq. 3 is defined positively using its low bound of Eq. 17: The potential energy has a low bound with $[1-\cos(x)] > \rho x^2 \longleftarrow |x| < \pi$:

$$V(\boldsymbol{q},\dot{\boldsymbol{q}}) \geq \frac{1}{2}\lambda_M^{min}||\dot{\boldsymbol{q}}||^2 + \rho\lambda_Q^{min}||\boldsymbol{q}||^2 + \epsilon\lambda_M^{max}||\boldsymbol{q}||||\dot{\boldsymbol{q}}|| > 0 \tag{17}$$

where $Q = \begin{bmatrix} g_1+g_2 & g_2 \\ g_2 & g_2 \end{bmatrix}$, λ_A^{min}, and λ_A^{max} represent the A matrix min and max eigenvalues. The conditions for Eq. 17 are the following:

$$\begin{aligned} \rho &> 0, \epsilon > 0 \\ \rho\lambda_Q^{min} &> 0 \frac{1}{2}\rho\lambda_Q^{min}\lambda_M^{min} - \frac{1}{4}(\epsilon\lambda_M^{max})^2 > 0 \\ ||\boldsymbol{q}|| &< \pi \end{aligned}$$

Eq. 18 shows the high bound of the derivate of Eq. 3.

$$\dot{V}(\boldsymbol{q},\dot{\boldsymbol{q}}) \leq \dot{\boldsymbol{q}}^{\boldsymbol{T}}\boldsymbol{\tau_{PD}} + \epsilon\boldsymbol{q}^{\boldsymbol{T}}\boldsymbol{\tau}_{PD} - [||\boldsymbol{q}||,||\dot{\boldsymbol{q}}||]R_1[||\boldsymbol{q}||,||\dot{\boldsymbol{q}}||]^{\boldsymbol{T}} \tag{18}$$

where

$$\begin{aligned} R_1 &= \begin{bmatrix} \rho\epsilon\lambda_Q^{min} & \frac{1}{2}\epsilon(\lambda_B^{max}+\lambda_K^{max}) \\ \frac{1}{2}\epsilon(\lambda_B^{max}+\lambda_K^{max}) & \lambda_H^{min} - \epsilon\lambda_M^{max} + \gamma\lambda\lambda_K^{min} \end{bmatrix} \\ H(\boldsymbol{q}) &= \begin{bmatrix} b_1 & -\frac{1}{4}\epsilon I_2 q_2 \sin(q_2) \\ -\frac{1}{4}\epsilon I_2 q_2\sin(q_2) & b_2 + \frac{1}{2}\epsilon I_2 q_2 \sin(q_2) \end{bmatrix} \\ & \dot{\boldsymbol{q}}^T[\dot{M}(\boldsymbol{q}) - 2C(\boldsymbol{q},\dot{\boldsymbol{q}})]\dot{\boldsymbol{q}} = 0 \\ \tanh(\lambda x)x &\geq \gamma\lambda x^2 \\ \boldsymbol{q^T g(q)} &\geq \boldsymbol{\rho q^T Q q} \end{aligned}$$

The following conditions are necessary to demonstrate that Eq. 18 is negative defined:

$$\begin{aligned} &\rho\epsilon\lambda_Q^{min}(\lambda_H^{min} - \epsilon\lambda_M^{max} + \gamma\lambda\lambda_K^{min}) \\ &> \frac{1}{4}\epsilon^2(\lambda_B^{max}+\lambda_K^{max})^2 \end{aligned}$$

The Eq. 18 is rewritten by adding a zero with an R_2 matrix such that $\lambda_{R_1}^{min} > \lambda_{R_2}^{max}$.

$$\begin{aligned} \boldsymbol{X} &= [||\boldsymbol{q}||,||\dot{\boldsymbol{q}}||]^T \\ \dot{V}(\boldsymbol{X}) &\leq -\boldsymbol{X}^T(R_1-R_2)\boldsymbol{X} + \dot{\boldsymbol{q}}^T\boldsymbol{\tau_{PD}} + \epsilon\boldsymbol{q}^T\boldsymbol{\tau_{PD}} - \\ \boldsymbol{X}^T R_2\boldsymbol{X} &< 0 \end{aligned} \tag{19}$$

The term $-\boldsymbol{X}^T(R_1-R_2)\boldsymbol{X}$ is defined negatively. Therefore the remaining terms hold the following inequality:

$$\sqrt{||\boldsymbol{q}||^2 + ||\dot{\boldsymbol{q}}||} > \frac{\sqrt{2}}{\lambda_{R_2}^{min}}||\boldsymbol{\tau_{PD}}|| \tag{20}$$

The high bound g_f of the input torque $\boldsymbol{\tau_{PD}}$ is made with the high and low bound of Eq. 3, and the function $P(\boldsymbol{\tau_{PD}}) = \frac{\sqrt{2}}{\lambda_{R_2}^{min}}||\boldsymbol{\tau_{PD}}||$:

$$g_f = \frac{\sqrt{2\lambda_{S_2}^{max}}}{\sqrt{\lambda_{S_1}^{min}\lambda_{R_2}^{min}}}||\boldsymbol{\tau_{PD}}|| \tag{21}$$

where

$$S_1 = \begin{bmatrix} \rho\lambda_Q^{min} & \frac{1}{2}\epsilon\lambda_M^{max} \\ \frac{1}{2}\epsilon\lambda_M^{max} & \lambda_M^{min} \end{bmatrix}, S_2 = \begin{bmatrix} \rho\lambda_Q^{max} & \frac{1}{2}\epsilon\lambda_M^{min} \\ \frac{1}{2}\epsilon\lambda_M^{min} & \lambda_M^{max} \end{bmatrix}$$

Finally, the ISS analysis demonstrates that the initial conditions and the input bounds the state vector $\boldsymbol{x} = [\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T]^T$, as Eq. 22 shows.

$$\begin{aligned} ||\boldsymbol{x}|| \leq\ & W(||\boldsymbol{x}(t_0)||, t-t_0) \\ & + a_1\left(\sup_{t_0\leq\mu\leq t} ||\boldsymbol{\tau_{PD}}||(\mu)\right) \end{aligned} \tag{22}$$

where $a_1 = \sqrt{2\lambda_{S_2}^{max}}/\sqrt{\lambda_{S_1}^{min}\lambda_{R_1}^{min}}$, $\frac{\partial}{\partial y_1}W(y_1,y_2) > 0$, and $\frac{\partial}{\partial y_2}W(y_1,y_2) < 0$. The parameters identified with the **algorithm 1** met all the conditions for stability with $\rho = 0.1$, $\epsilon = 0.001$, $\gamma = 0.0025$:

$$\begin{aligned} \rho\lambda_Q^{min} &= 0.0072 > 0 \\ & \frac{1}{2}\rho\lambda_Q^{min}\lambda_M^{min} - \frac{1}{4}(\epsilon\lambda_M^{max})^2 \\ &= 6.2020\times 10^{-5} > 0 \\ & \rho\epsilon\lambda_Q^{min}(\lambda_H^{min} - \epsilon\lambda_M^{max} + \gamma\lambda\lambda_K^{min}) \\ &= 1.3659\times 10^{-7} \\ &> \frac{1}{4}\epsilon^2(\lambda_B^{max}+\lambda_K^{max})^2 \\ &= 7.1236\times 10^{-8} R_2 = \rho\epsilon R_1 \\ \lambda_{R_1}^{min} &= 3.4422\times 10^{-6} \\ &> \lambda_{R_2}^{max} = 1.8986\times 10^{-6} \end{aligned}$$

REFERENCES

[1] G. I. Zamora-Gómez, A. Zavala-Río, D. J. López-Araujo, and V. Santibánez, "Further results on the global continuous control for finite-time and exponential stabilisation of constrained-input mechanical systems: Desired conservative-force compensation and experiments," *IET Control Theory Appl.*, vol. 13, no. 2, pp. 159–170, Jan. 2019, doi: 10.1049/iet-cta.2018.5099.

[2] S. Haddadin, A. De Luca, and A. Albu-Schäffer, ''Robot collisions: A survey on detection, isolation, and identification,'' *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1292–1312, Dec. 2017, doi: 10.1109/TRO.2017.2723903.

[3] J. Swevers, C. Ganseman, D. B. Tukel, J. de Schutter, and H. Van Brussel, ''Optimal robot excitation and identification,'' *IEEE Trans. Robot. Autom.*, vol. 13, no. 5, pp. 730–740, Oct. 1997, doi: 10.1109/70.631234.

[4] S. Zhang, S. Wang, F. Jing, and M. Tan, ''Parameter estimation survey for multi-joint robot dynamic calibration case study,'' *Sci. China Inf. Sci.*, vol. 62, no. 10, pp. 1–15, Aug. 2019, doi: 10.1007/s11432-018-9726-3.

[5] J. Moreno-Valenzuela, R. Miranda-Colorado, and C. Aguilar-Avelar, ''A MATLAB-based identification procedure applied to a two-degrees-of-freedom robot manipulator for engineering students,'' *Int. J. Electr. Eng. Educ.*, vol. 54, no. 4, pp. 319–340, Jan. 2017, doi: 10.1177/0020720916689102.

[6] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, ''A survey of convolutional neural networks: Analysis, applications, and prospects,'' *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 10, 2021, doi: 10.1109/TNNLS.2021.3084827.

[7] J. S. Albus, ''A new approach to manipulator control: The cerebellar model articulation controller (CMAC),'' *J. Dyn. Syst., Meas., Control*, vol. 97, no. 3, pp. 220–227, Sep. 1975, doi: 10.1115/1.3426922.

[8] R. T. Wu and M. R. Jahanshahi, ''Deep convolutional neural network for structural dynamic response estimation and system identification,'' *J. Eng. Mech.*, vol. 145, no. 1, pp. 1–25, Jan. 2019, doi: 10.1061/(ASCE)EM.1943-7889.0001556.

[9] H. Su, W. Qi, Y. Hu, J. Sandoval, L. Zhang, Y. Schmirander, G. Chen, A. Aliverti, A. Knoll, G. Ferrigno, and E. De Momi, ''Towards model-free tool dynamic identification and calibration using multi-layer neural network,'' *Sensors*, vol. 19, no. 17, pp. 1–18, Jul. 2019, doi: 10.3390/s19173636.

[10] H. Su, W. Qi, C. Yang, J. Sandoval, G. Ferrigno, and E. D. Momi, ''Deep neural network approach in robot tool dynamics identification for bilateral teleoperation,'' *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2943–2949, Apr. 2020, doi: 10.1109/LRA.2020.2974445.

[11] S. Liu, L. Wang, and X. V. Wang, ''Sensorless force estimation for industrial robots using disturbance observer and neural learning of friction approximation,'' *Robot. Comput.-Integr. Manuf.*, vol. 71, pp. 1–11, Oct. 2021, doi: 10.1016/j.rcim.2021.102168.

[12] C. Urrea and J. Pascal, ''Design and validation of a dynamic parameter identification model for industrial manipulator robots,'' *Arch. Appl. Mech.*, vol. 91, no. 5, pp. 1981–2007, Jan. 2021, doi: 10.1007/s00419-020-01865-2.

[13] O. F. Argin and Z. Y. Bayraktaroglu, ''Consistent dynamic model identification of the Stäubli RX-160 industrial robot using convex optimization method,'' *J. Mech. Sci. Technol.*, vol. 35, no. 5, pp. 2185–2195, Apr. 2021, doi: 10.1007/s12206-021-0435-1.

[14] N. Shao, Q. Zhou, C. Shao, and Y. Zhao, ''Adaptive control of robot series elastic drive joint based on optimized radial basis function neural network,'' *Int. J. Social Robot.*, vol. 13, no. 7, pp. 1823–1832, Mar. 2021, doi: 10.1007/s12369-021-00762-0.

[15] N. Liu, L. Li, B. Hao, L. Yang, T. Hu, T. Xue, S. Wang, and X. Shao, ''Semiparametric deep learning manipulator inverse dynamics modeling method for smart city and industrial applications,'' *Complexity*, vol. 2020, pp. 1–11, Jun. 2020, doi: 10.1155/2020/9053715.

[16] E. Madsen, S. A. Timm, N. A. Ujfalusi, O. S. Rosenlund, D. Brandt, and X. Zhang, ''Dynamics parametrization and calibration of flexible-joint collaborative industrial robot manipulators,'' *Math. Problems Eng.*, vol. 2020, pp. 1–13, Sep. 2020, doi: 10.1155/2020/8709870.

[17] T. Xu, J. Fan, Y. Chen, X. Ng, M. H. Ang, Q. Fang, Y. Zhu, and J. Zhao, ''Dynamic identification of the KUKA LBR iiwa robot with retrieval of physical parameters using global optimization,'' *IEEE Access*, vol. 8, pp. 108018–108031, 2020, doi: 10.1109/ACCESS.2020.3000997.

[18] S. Wang, X. Shao, L. Yang, and N. Liu, ''Deep learning aided dynamic parameter identification of 6-DOF robot manipulators,'' *IEEE Access*, vol. 8, pp. 138102–138116, 2020, doi: 10.1109/ACCESS.2020.3012196.

[19] G. Liu, Q. Li, L. Fang, B. Han, and H. Zhang, ''A new joint friction model for parameter identification and sensor-less hand guiding in industrial robots,'' *Ind. Robot, Int. J. Robot. Res. Appl.*, vol. 47, no. 6, pp. 847–857, Jul. 2020, doi: 10.1108/IR-03-2020-0053.

[20] P. Cao, Y. Gan, and X. Dai, ''Model-based sensorless robot collision detection under model uncertainties with a fast dynamics identification,'' *Int. J. Adv. Robot. Syst.*, vol. 16, no. 3, pp. 1–15, Jun. 2019, doi: 10.1177/1729881419853713.

[21] L. Zhang, J. Wang, J. Chen, K. Chen, B. Lin, and F. Xu, ''Dynamic modeling for a 6-DOF robot manipulator based on a centrosymmetric static friction model and whale genetic optimization algorithm,'' *Adv. Eng. Softw.*, vol. 135, pp. 1–9, Sep. 2019, doi: 10.1016/j.advengsoft.2019.05.006.

[22] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, ''Dynamic identification of the Franka Emika panda robot with retrieval of feasible parameters using penalty-based optimization,'' *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4147–4154, Oct. 2019, doi: 10.1109/LRA.2019.2931248.

[23] H. Ni, C. Zhang, T. Hu, T. Wang, Q. Chen, and C. Chen, ''A dynamic parameter identification method of industrial robots considering joint elasticity,'' *Int. J. Adv. Robot. Syst.*, vol. 16, no. 1, pp. 1–11, Feb. 2019, doi: 10.1177/1729881418825217.

[24] B. Yao, Z. Zhou, L. Wang, W. Xu, and Q. Liu, ''Sensor-less external force detection for industrial manipulators to facilitate physical human–robot interaction,'' *J. Mech. Sci. Technol.*, vol. 32, no. 10, pp. 4909–4923, Oct. 2018, doi: 10.1007/s12206-018-0939-5.

[25] R. Kumar, S. Srivastava, and J. R. P. Gupta, ''Online modeling and adaptive control of robotic manipulators using Gaussian radial basis function networks,'' *Neural Comput. Appl.*, vol. 30, no. 1, pp. 223–239, Jul. 2018, doi: 10.1007/s00521-016-2695-8.

[26] R. Miranda-Colorado and J. Moreno-Valenzuela, ''Experimental parameter identification of flexible joint robot manipulators,'' *Robotica*, vol. 36, no. 3, pp. 313–332, Mar. 2018, doi: 10.1017/S0263574717000224.

[27] J. Hu and R. Xiong, ''Contact force estimation for robot manipulator using semiparametric model and disturbance Kalman filter,'' *IEEE Trans. Ind. Electron.*, vol. 65, no. 4, pp. 3365–3375, Apr. 2018, doi: 10.1109/TIE.2017.2748056.

[28] S. Jiang, M. Jiang, Y. Cao, D. Hua, H. Wu, Y. Ding, and B. Chen, ''A typical dynamic parameter identification method of 6-degree-of-freedom industrial robot,'' *Proc. Inst. Mech. Eng. I, J. Syst. Control Eng.*, vol. 231, no. 9, pp. 740–752, Aug. 2017, doi: 10.1177/0959651817726477.

[29] C. Duan and R. Singh, ''Dynamics of a 3dof torsional system with a dry friction controlled path,'' *J. Sound Vib.*, vol. 289, nos. 4–5, pp. 657–688, Feb. 2006, doi: 10.1016/j.jsv.2005.02.029.

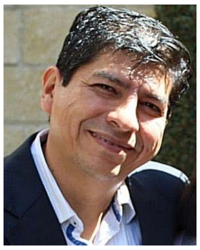
[30] R. Kelly and V. Santibáñez, *Control de Movimiento de Robots Manipuladores*, 1st ed. Madrid, Spain: Prentice-Hall, 2003.

[31] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002, pp. 174–180.

[32] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete Time Signal Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999, pp. 594–595.

[33] F. Berzal, *Redes Neuronales & Deep Learning*, 1st ed. Granada, Spain: Independent, 2018, pp. 599–686.

CARLOS LEOPOLDO CARREÓN-DÍAZ DE LEÓN was born in Mexico, in 1995. He received the B.S. degree in mechatronics engineering from the Universidad Veracruzana (UV), in 2017, and the M.S. degree in sciences of the electronics in automation from the Benemérita Universidad Autónoma de Puebla (FCE-BUAP), Mexico, in 2019. He is currently pursuing the Ph.D. degree in language and knowledge engineering in human–computer interaction and robotics research line. He has published two articles in international journals and one patent in process. His current research interests include robot model identification with neural networks, embedded soft core processors on FPGA, mechatronic systems, and robot signal processing.



SERGIO VERGARA-LIMÓN was born in Mexico, in 1970. He received the Ph.D. degree in optoelectronics science from the Faculty of Physics and Mathematics Sciences (FCFM), Benemérita Universidad Autónoma de Puebla (BUAP), in 2000. He is currently a full-time Professor-Researcher with the Faculty of Electronic Science (FCE), BUAP. He has published 168 articles in international journals and has six patents. His current research interest includes developing an automatic system based on FPGA architecture for different application, such as high-energy physics, quantum optics, control robotics, mechatronics, and automation.

MARÍA AURORA D. VARGAS-TREVIÑO was born in Mexico, in 1972. She received the Ph.D. degree in optoelectronics science from the Faculty of Physics and Mathematics Sciences (FCFM), Benemérita Universidad Autónoma de Puebla (BUAP), in 2000. She is currently a full-time Professor-Researcher with the Faculty of Electronic Science (FCE), BUAP. She has published 168 articles in international journals and has five patents. Her current research interest includes developing an automatic system based on FPGA architecture for different application, such as high-energy physics, quantum optics, control robotics, mechatronics, and automation.

JESÚS LÓPEZ-GÓMEZ was born in Tabasco, Mexico, in 1988. He received the B.S. degree in mechatronics engineering from the Tecnológico Nacional de México (ITSC), in 2011, the M.S. degree in sciences of the electronics in automation from the Benemérita Universidad Autónoma de Puebla (FCE-BUAP), Mexico, in 2015, and the Ph.D. degree in robotic and mechatronic systems engineering from the Instituto Politécnico Nacional (ESIME-IPN), Mexico, in 2019. He is currently a full-time Professor-Researcher. He has published ten articles in international journals and two patents in process. His current research interests include modeled and control of robotic and mechatronic systems, development of control systems in FPGA-based hardware/software, and artificial intelligence applications.

JUAN MANUEL GONZALEZ-CALLEROS was born in Mexico, in 1978. He received the Ph.D. degree in economic science and administration from the Institut d'Administration et de Gestion (IAG), Université Catholique de Louvain, in 2006. He is currently a full-time Professor-Researcher with the Faculty of Electronic Science (FCE), Benemérita Universidad Autónoma de Puebla (BUAP). He is currently a full-time Professor-Researcher with the Faculty of Computation Science (FCC), BUAP. He has published 52 articles in international journals and nine books. His current research interests include educational technology, software engineering, and human–computer iteration.

DANIEL MARCELO GONZÁLEZ-ARRIAGA was born in Mexico, in 1993. He received the master's degree in electronic sciences from the Faculty of Electronic Sciences (FCE), Benemérita Universidad Autónoma de Puebla (BUAP), in 2019. He is currently pursuing the Ph.D. degree in language and knowledge engineering with the Faculty of Computer Science (FCC), BUAP. He has published one article and has a copyright. His current research interest includes the development of neural network-based technologies for application in various areas.

MARCIANO VARGAS-TREVIÑO was born in Mexico, in 1974. He received the Ph.D. degree in physics instrumentation from the Université Joseph Fourier, Grenoble, France, in 2005. He is currently a full-time Professor-Researcher with the School of Biological Systems and Technological Innovations (SBIT), Universidad Autónoma Benito Juarez de Oaxaca (UABJO). He has published 17 articles in international journals and has one patent. His current research interests include developing medical instrumentation and applied biotechnology.