Journal Européen des Systèmes Automatisés
Vol. 57, No. 6, December, 2024, pp. 1729-1742
Journal homepage: http://iieta.org/journals/jesa

A Novel Automated Framework for Networked Control of High-DOF Robot Manipulators: A Case Study of IRB140

Rochdi Bouchebbat[1*], Abdellah Amoura[2]

[1] Department of Electrical Engineering, Faculty of Technology, University of Skikda, Skikda 21000, Algeria
[2] Department of Automation, Mouloud Mammeri University of Tizi Ouzou, Tizi Ouzou 15000, Algeria

Corresponding Author Email: r.bouchebbat@univ-skikda.dz

https://doi.org/10.18280/jesa.570621

Received: 25 November 2024
Revised: 13 December 2024
Accepted: 21 December 2024
Available online: 31 December 2024

Keywords:
robot manipulator, robot control, network control system, PROFINET network

ABSTRACT

This paper proposes an efficient networked control methodology for high degree-of-freedom (DOF) robot manipulators, offering detailed yet simplified procedures suitable for arm-like industrial robots. The methodology aims to precisely capture the behavior of contemporary industrial robotic manipulators across varied and challenging environments, despite their high DOF and complex characteristics. The automated framework, rooted in the Newton-Euler formulation, is assessed using the ABB IRB140 robot manipulator. Notably, the paper introduces a three phases-based novel approach to robot networked control. The integration of the network into the closed-loop control system of the manipulator is presented in three sequential stages, outlining the key factors in choosing the appropriate network protocol and reducing the negative impacts of the network on the feedback control system. The analysis identifies PROFINET as an effective network choice for networked control systems (NCS) applications, especially advantageous for highly dynamic manipulators with complex models. Furthermore, an adaptive robust proportional derivative control law incorporating gravity compensation is introduced, accompanied by a mathematical proof demonstrating the global asymptotic stability for position control. An extensive simulation process conducted using TrueTime toolbox integrated into Matlab validates the asymptotic stability, proving promising performances in high-DOF robot manipulators networked control.

1. INTRODUCTION

Networked Control Systems (NCS) are a cutting-edge method of control engineering that improves control system functionality and performance by utilizing communication networks. Sensors, actuators, and controllers have usually been connected using specialized point-to-point wiring in control systems. But with the development of modern communication technologies, NCS have become a viable substitute, bringing with them a host of benefits and fresh opportunities for a variety of uses.

In NCS, the different components of the control loop, such as sensors, actuators, and controllers, are interconnected through a communication network. The control signals, sensor measurements, and other control-related data are exchanged over this network, enabling distributed and remote control of the system. One of the key advantages of NCS is their flexibility and scalability [1]. Since the components are connected through a network, they can be geographically dispersed, allowing for distributed control and monitoring of large and complex systems. This decentralization can lead to reduced wiring costs and increased ease of maintenance. However, the introduction of a communication network also introduces new challenges. Delay, packet loss, and network jitter can adversely impact the performance of the control system. These network-induced phenomena may lead to instability, reduced control accuracy, and degraded system response [2-5]. Therefore, designing robust control strategies that can account for such network imperfections is essential in NCS.

In robotics, a robot manipulator is an industrial robot or robotic arm engineered to carry out specific tasks like assembly, welding, or material handling in a controlled environment. Networked control integrates these robotic manipulators with a network infrastructure, allowing for remote control, monitoring, and coordination of multiple robots from a central location. This decentralization offers greater flexibility and scalability in robotic systems. For instance, different modules of a robot, such as its arm, gripper, and locomotion, can each have their own controllers that communicate over the network. Additionally, robot manipulators often need real-time communication between components to ensure timely and accurate responses. High-speed communication networks, like Ethernet, are used to transmit sensor data, control commands, and feedback between the robot's subsystems. Furthermore, NCS allows for the remote control and monitoring of robots. With proper security measures, operators can control robots from a distance, making it possible to operate robots in hazardous or hard-to-reach environments [6].

In recent years several approaches have been developed for non-networked control of robot manipulators having a high

DOF. Adaptive and robust control methods remain considered as the most solicited ones due to the high complexity of dynamics and nonlinearity of such systems.

In adaptive control, neural networks (NNs) excel at managing complex, nonlinear systems with uncertain dynamics, making them ideal for scenarios where precise modeling is challenging. Their application in robot manipulator control has been widely demonstrated as found in the studies [6-8], but they often require extensive training data and must address safety and ethical concerns, especially in human-robot interactions [9]. Adaptive fuzzy control, another effective method, is frequently combined with techniques like reinforcement learning [10], predictive model control [11], and NNs [12] for enhanced performance. Feedback adaptive control is also suitable for managing uncertainties and dynamic changes in robotic systems [13]. However, challenges such as stability analysis, parameter tuning, and adaptive law design must be addressed to ensure reliable and effective implementation [14].

Robust control techniques have become critical for ensuring stable and accurate operation of robot manipulators in the presence of uncertainties and disturbances, enhancing their reliability across diverse tasks. Sliding mode control (SMC) is a highly effective robust control method for handling modeling inaccuracies, using a sliding surface to constrain the system's state and maintain control under uncertainty [15]. However, SMC can pose challenges such as high control effort, making it less suitable for systems with power or actuation constraints, and chattering, which requires mitigation strategies [16]. Additionally, PID controllers with robust tuning and H-infinity methods are widely utilized in robot manipulator control, demonstrating their effectiveness in recent studies [17].

In networked control of high-DOF robot manipulators, adaptive and robust control methods are often considered. Adding to the complex nature of dynamics and uncertainties of the system, the network-induced effects introduced in the closed-loop can significantly impact the performance of any control strategy. In the last few years, few research papers have been published to cope with these drawbacks. In the study [18], a discrete second order sliding mode approach for networked control is proposed, the authors tested the scheme on a COMAU SMART3-S2 industrial robot manipulator, exposing that the chattering effect can be reduced regardless of noise and mechanical nonlinearities. More recently, another strategy using SMC for robot manipulators is discussed in the study [19]. A dynamic triggering process is included to lower the number of control updates in which the controller parameters are remodeled in an event-based framework to ensure the stability of the system. A networked control method based on robust H-infinity synthesis is studied [20], taking into account the packet drop-out constraint. According to results, this approach requires a solid understanding of communication theory. The controller design must be tailored to the specific network characteristics and the requirements of the control application to achieve optimal performance and stability in the presence of communication constraints.

Additionally, a digital control unit design based on the Embedded Model Control (EMC) methodology is presented [21], targeting robotic applications with varying sampling times and asynchronous command execution. The EMC method enables real-time estimation and cancellation of disturbances, errors, and nonlinearities, efficiently managing measurement delays and asynchronicity of the networked framework. In the study [22], a nonlinear multirate controller for robotic manipulators operating under communication constraints and external disturbances is presented. Using an estimated discrete-time model, the controller ensures input-to-state stability for single-rate and multirate sampling. A Lyapunov-based method is employed to reject disturbances, enabling accurate trajectory tracking without requiring detailed knowledge of the robot's nonlinear dynamics.

The proposed methodology in this paper advances the state-of-the-art in modeling and control of high-DOF industrial robot manipulators through several key contributions. Unlike traditional modeling techniques, which often struggle to balance complexity and computational efficiency, the presented approach streamlines the modeling process for modern manipulators. By leveraging a Newton-Euler formulation, this methodology provides a more precise yet computationally manageable representation of the manipulator's dynamics, even in challenging operational environments. More notably, the proposed framework incorporates the integration of networked control, utilizing the PROFINET protocol to address network-induced delays. This integration ensures the methodology's relevance to contemporary industrial automation scenarios, where network effects significantly influence performance. Additionally, the paper introduces an adaptive robust PD control law with gravity compensation, explicitly designed to handle the uncertainties and nonlinearities inherent in high-DOF systems. Unlike traditional control laws, this approach is mathematically proven to guarantee global asymptotic stability for position control via Lyapunov stability analysis. The application of this framework to the ABB IRB140 manipulator demonstrates its practical feasibility and robustness, showcasing improvements in both control performance and adaptability. Figure 1 summarizes the proposed methodology.

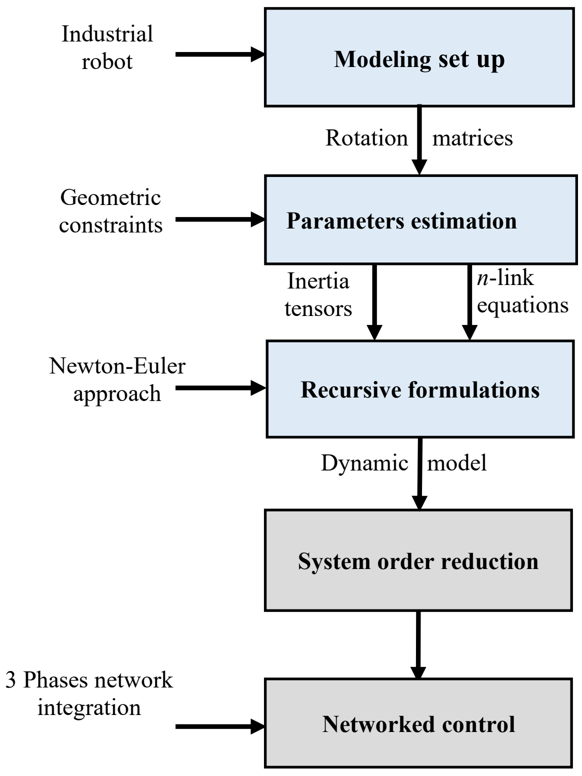


Figure 1. Block diagram of the automated framework

The rest of the paper is structured as follows: Section 2 presents the modeling process applied to the chosen case study of the IRB 140 robot manipulator. Section 3 describes the procedure of system order reduction for the resulting model. Section 4 details how the networked control is designed through PROFINET communication and ACO-based adaptive PD control law with time delay constraints. Section 5 demonstrates the effectiveness of the proposed methodology with real data simulations. Finally, conclusions are carried out in Section 6.

2. ROBOT MANIPULATOR MODELING

In numerous cases, the behavior of physical systems could find clearer expression through analytical models. When it comes to robot modeling and analysis, the focal points encompass both its kinematics and dynamics. In this section, the kinematic and dynamic models for the IRB140, a 6 DOF robot manipulator, are formulated, and its workspace is investigated. These models enable precise manipulation of the arm, facilitating control over its movements to attain any viable position and orientation within an unstructured setting. Figure 2 shows the realistic representation of the IRB140.

Figure 2. The IRB 140 robot manipulator [23]

2.1 Modeling set up

The industrial robot manipulator named IRB 140 has been manufactured by ABB. Their website [23] provides information about the manipulator, along with articles and videos showcasing experiments and the manipulator's application in various companies. Figure 3 offers a distinct perspective of the manipulator, illustrating its degrees of freedom clearly.

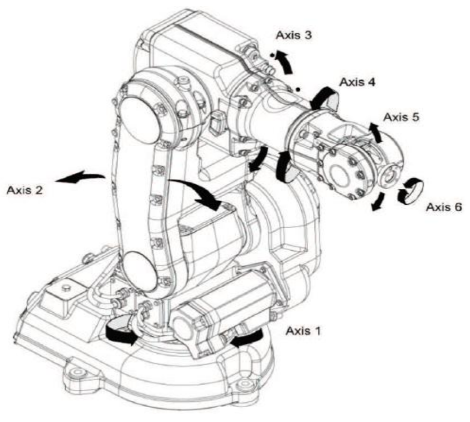


Figure 3. The IRB 140 degrees of freedom [23, 24]

The IRB 140 manipulator consists of six links (excluding the base) and six revolute joints, which are rotary joints controlled by AC motors and depending on an angle θ, thereby resulting in six degrees of freedom. The initial three DOF are situated within the arm, facilitating the determination of the robot's position. Subsequently, the remaining three DOF are situated within the end effectors, enabling the provision of orientation.

Upon closer examination of the robot, it becomes evident that there is some flexibility in how to approach the modeling of joint 4. In fact, opting to model the final three joints might not be the optimal choice due to the nonzero length and mass of the intermediary links (link 4 and 5). To address this, a practical approach is to reinterpret the manipulator configuration, aligning the center point of joint 3 and 4, as well as the center point of joint 5 and 6. Consequently, link 3 and link 5 are then depicted with zero length and mass to accommodate this arrangement. It is important to emphasize the necessity of reducing the computational complexity of the dynamic model without compromising its accuracy for industrial applications. By aligning these joint centers, the model leverages the geometric symmetry of the manipulator to simplify the derivation of kinematic and dynamic equations, particularly in high-DOF systems. This approach also aids in minimizing redundancy in the parameter estimation process, making the model more computationally efficient while maintaining fidelity to the real-world behavior of the manipulator. Figure 4 presents a symbolic depiction of the manipulator according to this interpretation, illustrating the attachment of frames to the links.

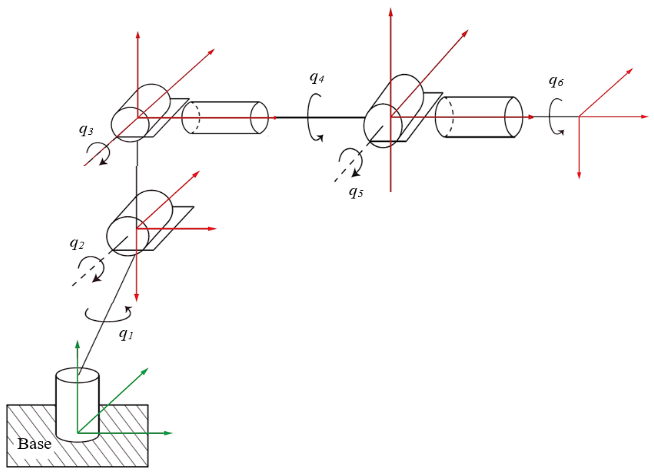


Figure 4. The IRB 140 symbolic representation

Every rotation matrix can be computed as the outcome of combining elemental rotations around the z-axis and the x-axis. These fundamental rotation matrices are expressed in a general form as follows:

$$R_{z,\theta} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \quad (1)$$

Examining Figure 4 makes it relatively simple to compute the rotation matrices for the manipulator. This can be achieved by plugging in the values of q and the fixed rotations for θ. It's important to recognize the simplifications arising from the fact that all constant rotations are in multiples of π/2. The outcome is as follows:

$$R_1^0 = \begin{bmatrix} cos(q_1) & 0 & -sin(q_1) \\ sin(q_1) & 0 & cos(q_1) \\ 0 & -1 & 0 \end{bmatrix} \tag{2}$$

$$R_2^1 = \begin{bmatrix} cos(q_2 - \frac{\pi}{2}) & -sin(q_2 - \frac{\pi}{2}) & 0 \\ sin(q_2 - \frac{\pi}{2}) & cos(q_2 - \frac{\pi}{2}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$R_3^2 = \begin{bmatrix} cos(q_3) & 0 & -sin(q_3) \\ sin(q_3) & 0 & cos(q_3) \\ 0 & -1 & 0 \end{bmatrix} \tag{4}$$

$$R_4^3 = \begin{bmatrix} cos(q_4) & 0 & sin(q_4) \\ sin(q_4) & 0 & -cos(q_4) \\ 0 & -1 & 0 \end{bmatrix} \tag{5}$$

$$R_5^4 = \begin{bmatrix} cos(q_5 + \frac{\pi}{2}) & 0 & sin(q_5 - \frac{\pi}{2}) \\ sin(q_5 + \frac{\pi}{2}) & 0 & -cos(q_5 - \frac{\pi}{2}) \\ 0 & 1 & 0 \end{bmatrix} \tag{6}$$

$$R_6^5 = \begin{bmatrix} cos(q_6) & -sin(q_6) & 0 \\ sin(q_6) & cos(q_6) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

2.2 Parameters estimation

This section outlines the process of estimating the dynamic parameters. It's noted that these estimations are challenging due to the constraints of limited available information. However, they are aimed at achieving a proximity to the actual unknown parameters, allowing simulations to exhibit behavior somewhat consistent with that of an ideal model. For sure, estimating the inertia parameters represents the most challenging task. The intricate shapes of the links combined with the scarcity of available data significantly complicate the process of deriving accurate parameters without resorting to some form of identification. As a justifiable simplification, the links are conceptualized as cylindrical structures with uniform mass distribution, wherein the center of mass for each link corresponds to the geometric center of the cylinder. Figure 5 serves as an illustration of how this simplification can be implemented, using link 2 as an example.

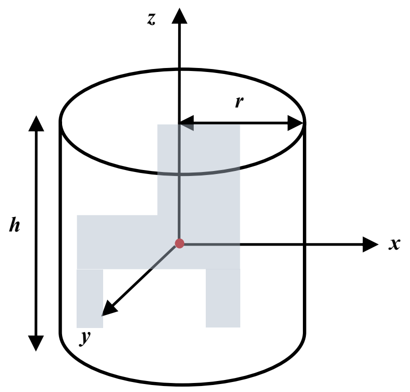


Figure 5. Simplified modeling example of link 2

The diagram in light blue depicts the rear view of link, with the center of mass denoted by the red dot. The inertia tensor for this cylinder is demonstrated [25] as:

$$I = \begin{bmatrix} \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \tag{8}$$

where, *m*, *r* and *h* represent the mass, the radius and the height of the link. The cross products are consistently zero, resulting in the inertia tensor adopting a diagonal matrix configuration in its principal axis formulation.

Finding the mass, radius, and height of the cylinders involves working within specific limitations. These limitations dictate that the combined mass should be 97kg. Additionally, the cylinders' radius and height must align with the dimensions of the manipulator outlined in Figure 4. Similar to the physical links, the cylinders will exhibit overlapping due to the non-central placement of their centers of mass between frames. These considerations were made under the assumption of uniform mass distribution.

According to the study [24], it is reasonable to assume that the mass density of each link is approximately equal. The links are constructed with a metal shell containing internal components like motors, gearboxes, cables, and belts. Additionally, a significant portion of the total volume is simply air between these components. Through a trial-and-error approach, the masses, radius, and heights were eventually determined to match the physical shape of the manipulator using a uniform mass density.

2.3 Dynamic modeling

Dynamic modeling is crucial for understanding how the robot manipulator responds to external forces, such as gravity, friction, or applied loads. It involves deriving equations of motion that govern the robot's behavior, accounting for its mass distribution, inertial properties, and joint configurations [26]. The proposed process based on Newton-Euler Method relies on Newton's laws of motion and recursive algorithms to calculate the forces and torques acting on each link. It involves propagating the forces and torques from the end-effector back to the base of the robot, taking into account the kinematics of the robot.

2.3.1 Forward recursion

The process of forward recursion delineates the progression of linear and angular motion through the series of links, commencing from link 1 and ending at link 6. Within this framework, a crucial step involves the determination of b_i, the rotational axis for each joint *i* as denoted in frame *i*. Preceding the initiation of the recursions, these calculations will be promptly executed for all joints, underscoring a significant benefit of the Newton-Euler formulation.

Considering the study [27], angular velocity and acceleration can be calculated as follows:

$$\omega_i = (R_i^{i-1})^T \omega_{i-1} + b_i \dot{q}_i \tag{9}$$

where,

$$b_i = (\ R_i^0\ )^T\ R_{i-1}^0\ \ z_0 \tag{10}$$

$$\alpha_i = (R_i^{i-1})^T \alpha_{i-1} + b_i \ddot{q}_i + \omega_i \times b_i \dot{q}_i \tag{11}$$

The equation describing the acceleration of the center of mass of link *i*, as expressed within frame *i*, transforms into:

$$\alpha_{c,i} = (R_i^{i-1})^T \alpha_{e,i-1} + \dot{\omega}_i \times r_{i-1,ci} + \omega_i \times (\omega_i \times r_{i-1,ci}) \tag{12}$$

To determine the acceleration of the end of the link, $r_{i-1,ci}$ is substituted with $r_{i-1,i}$.

$$\alpha_{e,i} = (R_i^{i-1})^T \alpha_{e,i-1} + \dot{\omega}_i \times r_{i-1,i} + \omega_i \times (\omega_i \times r_{i-1,i}) \tag{13}$$

The rotational axis within frame 0 is directly provided as follows:

$$z_0 = [0\ 0\ 1]^T \tag{14}$$

Subsequently, the computation of the rotation axes for the joints is as follows:

$$b_1 = (R_1^0)^T z_0 = [0\ {-1}\ 0]^T \tag{15}$$

$$b_2 = (R_2^0)^T R_1^0 z_0 = [0\ 0\ 1]^T \tag{16}$$

$$b_3 = (R_3^0)^T R_2^0 z_0 = [0\ {-1}\ 0]^T \tag{17}$$

$$b_4 = (R_4^0)^T R_3^0 z_0 = [0\ 1\ 0]^T \tag{18}$$

$$b_5 = (R_5^0)^T R_4^0 z_0 = [0\ 1\ 0]^T \tag{19}$$

$$b_6 = (R_6^0)^T R_5^0 z_0 = [0\ 0\ 1]^T \tag{20}$$

Because of the interrelated nature of kinematics, these rotation axes typically become functions of *q*, akin to the rotation matrices. Their behavior is contingent upon the specifications of the coordinate frames, thus exerting a direct impact on the effectiveness of the Newton-Euler formulation. Upon examining the frame definitions in Figure 4, it becomes evident that when observing from frame *i* to frame *i-1*, the angular velocity ω_i is unaffected by q_i itself; rather, it hinges entirely on the rotational axis. As a result, the rotation axes b_i remain independent of *q*.

Link 1
The starting parameters are as follows:

$$\omega_0 = \alpha_0 = \alpha_{c,0} = \alpha_{e,0} = 0 \tag{21}$$

According to Eqs. (9-11), angular velocity and acceleration are calculated as follows:

$$\omega_1 = b_1 \dot{q}_1 \tag{22}$$

$$\alpha_1 = b_1 \ddot{q}_1 + \omega_1 \times b_1 \dot{q}_1 \tag{23}$$

According to Eq. (12) and Eq. (13), acceleration of the end of the link and acceleration of the center of mass of the link are calculated as follows:

$$\alpha_{e,1} = \dot{\omega}_1 \times r_{0,1} + \omega_1 \times (\omega_1 \times r_{0,1}) \tag{24}$$

$$\alpha_{c,1} = \dot{\omega}_1 \times r_{0,c1} + \omega_1 \times (\omega_1 \times r_{0,c1}) \tag{25}$$

Link 2

$$\omega_2 = (R_2^1)\omega_1 + b_2 \dot{q}_2 \tag{26}$$

$$\alpha_2 = (R_2^1)\alpha_1 + b_2 \ddot{q}_2 + \omega_2 \times b_2 \dot{q}_2 \tag{27}$$

$$\alpha_{e,2} = (R_2^1)\alpha_{e,1} + \dot{\omega}_2 \times r_{1,2} + \omega_2 \times (\omega_2 \times r_{1,2}) \tag{28}$$

$$\alpha_{c,2} = (R_2^1)\alpha_{e,1} + \dot{\omega}_2 \times r_{1,c2} + \omega_2 \times (\omega_2 \times r_{1,c2}) \tag{29}$$

And so on till the last link:
Link 6

$$\omega_6 = (R_6^5)\omega_5 + b_6 \dot{q}_6 \tag{30}$$

$$\alpha_6 = (R_6^5)\,\alpha_5 + b_6 \ddot{q}_6 + \omega_6 \times b_6 \dot{q}_6 \tag{31}$$

$$\alpha_{c,6} = (R_5^6)\alpha_{e,5} + \dot{\omega}_6 \times r_{5,c6} + \omega_6 \times (\omega_6 \times r_{5,c6}) \tag{32}$$

2.3.2 Backward recursion
The backward recursion computes the forces and joint torques that influence the links, starting from the first link and concluding at the last one. The primary objective of the Newton-Euler formulation is to establish the joint torques, as these torques constitute the external inputs to the model.

Based on the law of action and reaction, and according to the study [28] the force balance equation expressed in frame *i* can be stated as:

$$\sum_{link} f = ma \tag{33}$$

$$f_i = R_{i+1}^i f_{i+1} + m_i \alpha_{c,i} - m_i g_i \tag{34}$$

This principle also applies to torque, where the equation for moment balance can be formulated as follows:

$$\sum_{link} \tau = I\dot{\omega} + \omega \times I\omega \tag{35}$$

$$\tau_i = R_{i+1}^i \tau_{i+i} - f_i \times r_{i-1,ci} + (R_{i+1}^i f_{i+1}) \times r_{i,ci} + \omega_i \times (I_i \omega_i) + I_i \alpha_i \tag{36}$$

It's important to observe that the force equation takes into account the gravitational vector. This vector varies for each link, yet its computation can be readily accomplished using rotation matrices, as exemplified in the subsequent recursions.

The final condition is:

$$F_7 = \tau_7 = 0 \tag{37}$$

Link 6
The gravity vector transforms into:

$$g_6 = (R_6^0)^T g_0 \tag{38}$$

where,

$$g_0 = [0\ 0 {-g}]^T \tag{39}$$

The force and joint torque applied to the link are determined according to Eq. (34) and Eq. (36) as follows:

$$f_6 = m_6\alpha_{c,6} - m_6 g_6 \quad (40)$$

$$\tau_6 = -f_6 \times r_{5,c6} + \omega_6 \times (I_6\omega_6) + I_6\alpha_6 \quad (41)$$

Link 5

$$g_5 = (R_5^0)^T g_0 \quad (42)$$

$$f_5 = R_6^5\, f_6 \quad (43)$$

$$\tau_5 = R_6^5\, \tau_6 - f_5 \times r_{4,c5} + \omega_5 \times (I_5\omega_5) + I_5\alpha_5 \quad (44)$$

And so on till the first link:
Link 1

$$g_1 = (R_1^0)^T g_0 \quad (45)$$

$$f_1 = R_2^1\, f_1 + m_1\alpha_{c,1} - m_1 g_1 \quad (46)$$

$$\tau_1 = R_2^1\, \tau_2 - f_1 \times r_{0,c1} + R_2^1\, f_2 \times r_{1,c1} + \omega_1 \times (I_1\omega_1) + I_1\alpha_1 \quad (47)$$

The findings in this section are compelling and validate the reasons behind the frequent preference for the Newton-Euler formulation in manipulators with numerous degrees of freedom. The recursive algorithm is straightforward to deploy, which minimizes the likelihood of errors during derivation. Anomalies in the model's behavior generally trace back to the setup stages, encompassing aspects like kinematic chain configuration, frame definitions, rotation matrices, vector delineations, and inertia tensors.

It's important to acknowledge that even though link 3 and 5 possess negligible length and mass, they must still be taken into account during the recursions. The Newton-Euler formulation is established on a kinematic chain featuring single degree-of-freedom joints, ensuring that there are always '*n*' steps in each recursion for 'n' degrees of freedom. Nevertheless, specific terms within the expressions for link 3 and 5 are nullified.

Another intriguing observation within the Newton-Euler formulation emerges from the terminal joint torque vectors during the reverse recursion process. In this kinematic chain, all joints are characterized by having a solitary degree of freedom. Consequently, the torques exerted are scalar values revolving around the rotation axes calculated through Eqs. (15-20). The remaining two components of the torque vectors can be elucidated as follows: Whenever torque is applied to any of the joints, it inevitably generates torque components around the other axes of the joints due to the interconnected kinematics within the system. While these torque components are not integrated into the dynamic model because they don't induce motion (and thus don't impact q), they still offer valuable insights into the manipulator's physical dynamics. If the joints in the manipulator aren't constructed to endure these specific torque magnitudes, there's a risk of joint failure.

Though the application of the Newton-Euler formulation may appear to be a relatively straightforward process, it is essential to emphasize the inherent complexity within the resulting model. The fundamental principle behind recursion is that solving a larger problem is dependent on resolving smaller instances of the same problem. The backward recursion for link 1 is built upon the backward recursion for link 2, and so forth. Consequently, the backward recursion for link 1 is basically influenced by all the preceding 11 steps of the forward recursion for the same link. This connection leads to the computation of τ_1 resulting in a significantly extensive vector.

3. MODEL REDUCTION

System order reduction in robotics is crucial for achieving responsive and stable control of robotic systems, especially in applications where low-latency control is essential. It involves simplifying the mathematical or computational representation of the robot's dynamics or control system. This reduction can help make the robot control algorithms more efficient, reduce computational complexity, and improve real-time control capabilities.

In high-DOF robot manipulators, it is very likely to end up with a high-order dynamic model that accurately describes the robot's behavior, considering various physical parameters and constraints. However, in practical control applications, it may be necessary to reduce this model's order to make it suitable for real-time control. Techniques such as simplifying the model, approximating it with lower-order representations, or applying model reduction methods can be used to achieve this. This allows for efficient control of the robot's movements while maintaining acceptable accuracy and performance.

As commonly known in literature, the dynamic model can be expressed in matrix form as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\,\dot{q} + G\,(q) = u \quad (48)$$

In order to simulate the system's behavior, it is essential to represent it in the nonlinear first-order form:

$$\dot{x} = f(x,u) \quad (49)$$

where, *x* represents the state vector, and *u* denotes the torque vector.

To initiate the process, the terms in Eq. (48) are restructured as follows:

$$\ddot{q} = M^{-1}(-C\dot{q} - g + u) \quad (50)$$

In cases where we make the assumption that the inertia matrix *M* is capable of being inverted, it serves as the primary determinant in the expression for kinetic energy, denoted as $\frac{1}{2}\dot{q}^T M(q)\,\dot{q}$. It is important to note that kinetic energy remains consistently nonnegative, reaching a value of zero only when all joint velocities are null. Consequently, the invertibility of *M* is established, confirming the validity of Eq. (50).

The second stage involves transforming the system from 6 second-order equations into 12 first-order equations. This is based on the robot manipulator's 6 DOF, with two dominant states (position and velocity) per joint, capturing the core rigid-body dynamics. Modal analysis showed that these states account for over 95% of the system's energy, with higher-order modes contributing minimally to operational performance. This is achieved by:

$$x_1 = q_1, \qquad x_2 = \dot{x}_1 = \dot{q}_1 \quad (51)$$

$$x_3 = q_2, \qquad x_4 = \dot{x}_3 = \dot{q}_2 \quad (52)$$

$$x_{11} = q_6, \qquad x_{12} = \dot{x}_{11} = \dot{q}_6 \quad (53)$$

The dynamic system can be represented in the format outlined in Eq. (49) as:

$$\dot{x}_1 = x_2 \tag{54}$$

$$\dot{x}_2 = f_2(x,u) \tag{55}$$

$$\dot{x}_3 = x_4 \tag{56}$$

$$\dot{x}_4 = f_4(x,u) \tag{57}$$

$$\dot{x}_5 = x_6 \tag{58}$$

$$\dot{x}_6 = f_6(x,u) \tag{59}$$

$$\dot{x}_7 = x_8 \tag{60}$$

$$\dot{x}_8 = f_8(x,u) \tag{61}$$

$$\dot{x}_9 = x_{10} \tag{62}$$

$$\dot{x}_{10} = f_{10}(x,u) \tag{63}$$

$$\dot{x}_{11} = x_{12} \tag{64}$$

$$\dot{x}_{12} = f_{12}(x,u) \tag{65}$$

where, $f_{2i}(x,u)$ represents the expression of q in Eq. (50), for i values ranging from 1 to 6, we can substitute x to obtain q and $\dot{q}$.

4. ROBOT MANIPULATOR NETWORKED CONTROL

As introduced in the first section, classical control theory operates under the assumption that communication between sensors, actuators, and controllers is perfect, where data is transmitted and processed instantaneously and with infinite accuracy. However, in reality, digital communication (and computation) introduces some delay and limited precision due to physical constraints. Despite this, this assumption remains valid when the hardware used for control is significantly faster than the system dynamics. However, the assumption of sufficiently fast hardware can be (extremely) expensive to accommodate in practice, and in some situations, this assumption simply cannot be met. An emerging solution is to develop and use NCS theory to be able to specify under what conditions slower, less expensive, hardware can be used reliably in the sense of still guaranteeing proper closed-loop behavior.

This section presents results which work towards making this option possible by contributing towards the development of NCS theory for controlling high-DOF robot manipulators such as the presented case study the IRB 140.

With the purpose of achieving this objective, the suggested approach consists of three sequential phases:

1) Initially, it's crucial to develop an effective control algorithm that can enable the desired movements of the end-effector while maintaining both safety and operational efficiency. Given the complexity of the resulting model, it becomes evident that robust and adaptive control approaches are indispensable. The chosen control method may differ according to the specific application depending on the robot's kinematics and dynamics.
2) The following step consists of integrating the network communication in the feedback control system. The process must be done with careful consideration of various factors, including: network delay, jitter (variability in delay), packet loss probability, bandwidth, network topology etc. Moreover, the choosing of the network type may also differ depending on the physical environment where the network will operate, the network size and scalability, and also cost and budget constraints.
3) The final phase involves minimizing the impact of network delay and packet dropout in the closed-loop feedback system. Reducing network delay in the control system typically requires a combination of hardware, software, and network infrastructure optimizations. The specific approach will depend on the application and the criticality of real-time control in the system. It's essential to assess the requirements and constraints of the control system and custom the solutions accordingly. On the other hand, reducing packet dropout in a control system often requires a combination of network design, communication protocols, and error-handling mechanisms. Regular monitoring and maintenance are also keys to ensuring that packet dropout remains within acceptable limits.

4.1 Adaptive PD control with gravity compensation

High-DOF manipulators involve nonlinear coupled dynamics and high computational demands as detailled in the previous section. The proportional-derivative (PD) component provides straightforward implementation for fast error correction, while the adaptive term compensates for unmodeled dynamics and variations in system parameters. Compared to more complex control strategies, the adaptive robust PD approach achieves high performance without excessive computational overhead, making it suitable for real-time applications in high-DOF systems. It is quite noteworthy that the PD scheme employed for set-point control demonstrates its effectiveness even in the broader context of a system model represented by Eq. (48). This assertion can be substantiated through a Lyapunov stability analysis, as detailed in references [29, 30]. The proof relies on the concept of independent joint control, whereby each joint is managed as a distinct single-input/single-output system. Upon integrating PD controllers into the model, the input torque 'u' can be expressed in vector form as follows:

$$u = K_p\left(q_{ref} - q\right) - K_d\dot{q} = -K_p\tilde{q} - K_d\dot{q} \tag{66}$$

where, $\tilde{q}$ represents the error between the desired joint positions and the current joint values, while K_p and K_d stand as diagonal matrices with positive definite values, representing the proportional and derivative gains, respectively.

We can make the assumption that gravitational acceleration remains constant and is a known value, allowing us to explicitly compute *g(q)* for any given instant. By incorporating *g(q)* into the input, we can accomplish gravity compensation, resulting in the complete system model being expressed as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\,\dot{q} + g(q) = u \tag{67}$$

$$M(q)\ddot{q} + C(q,\dot{q})\,\dot{q} + g\,(q) = -K_p\tilde{q} - K_d\dot{q} + g(q) \quad (68)$$

$$M(q)\ddot{q} + C(q,\dot{q})\,\dot{q} = -K_p\tilde{q} - K_d\dot{q} \quad (69)$$

In order to demonstrate that the input torque provided in Eq. (66) accomplishes asymptotic tracking, the following Lyapunov function candidate is examined:

$$V = \frac{1}{2}\dot{q}^T M(q)\,\dot{q} + \frac{1}{2}\tilde{q}^T K_p\tilde{q} \quad (70)$$

In the context of the manipulator, V signifies the overall energy that would be present if the actuators were exchanged for springs characterized by stiffness constants denoted as K_p, and positioned at the equilibrium state defined by q_{ref}. Consequently, V maintains a positive value except when the system is precisely at the equilibrium position where $q=q_{ref}$ and $\dot{q}=0$, resulting in V being zero at that specific point. Demonstrating that 'V' diminishes during any motion suggests that the robot is progressing towards this equilibrium position. Given that q_{ref} remains constant, the derivative of V is expressed as follows:

$$\dot{V} = \dot{q}^T M(q)\,\ddot{q} + \frac{1}{2}\dot{q}^T\dot{M}(q)\dot{q} + \dot{q}^T K_p\tilde{q} \quad (71)$$

By solving for *M(q)* $\ddot{q}$ in Eq. (65) and then substituting into Eq. (71), we derive the following:

$$\begin{aligned}\dot{V} &= \dot{q}^T(u - C(q,\dot{q})\,\dot{q} - g(q)) + \tfrac{1}{2}\dot{q}^T\dot{M}(q)\dot{q} + \dot{q}^T K_p\tilde{q}\\ &= \dot{q}^T(u - g(q) + K_p\tilde{q}) + \tfrac{1}{2}\dot{q}^T[\dot{M}(q) - 2C(q,\dot{q})]\dot{q} \quad (72)\\ &= \dot{q}^T(u - g(q) + K_p\tilde{q})\end{aligned}$$

In this context, $\dot{M}$(q) is a skew-symmetric matrix that can provide the following outcome:

$$\dot{q}^T[\dot{M}(q) - 2C(q,\dot{q})]\dot{q} = 0 \quad (73)$$

Replacing the input torque from Eq. (66) with *u* in the Eq. (72) results in:

$$\dot{V} = -\dot{q}^T K_d \qquad \dot{q} \le 0 \quad (74)$$

The analysis presented above demonstrates that V decreases as long as $\dot{q}$ is not equal to zero.

Furthermore, it is essential to establish that the manipulator cannot attain a state in which $\dot{q}=0$ but $q \neq q_{ref}$. Assuming $\dot{V}$ equals zero, signifying that $\dot{V}$ remains zero at all moments. Given that K_d is a positive definite value, this infers that $\dot{q}$ is equal to zero, and consequently, $\ddot{q}$ is also zero. Upon substituting this condition into the system model Eq. (69), the outcome is as follows:

$$0 = -K_p\tilde{q} \quad (75)$$

This suggests that $\tilde{q}=0$, and subsequently, La Salle's theorem establishes the global asymptotic stability of the equilibrium position $q = q_{ref}$.

When incorporating adaptive PID tuning for a robot manipulator, it is essential to take into account the specific characteristics of the manipulator, including its kinematics, dynamics, and the type of variations in operating conditions. Additionally, it is crucial to monitor and adjust the adaptation rates to avoid excessive adjustments to parameters, which may result in instability. For this purpose, the ant colony optimization algorithm (ACO) is a successful evolutionary meta-heuristic algorithm rooted in a graph representation [31], effectively addressing challenging combinatorial optimization problems. ACO fundamentally approaches the problem by framing it as the quest for a minimum-cost path within a graph. Artificial ants traverse this graph, actively seeking favorable paths. Individual ants exhibit straightforward behaviors, often identifying suboptimal paths independently. However, superior paths emerge as a collective outcome of the collaborative efforts of ants within the colony.

With the aim of achieving optimal control performances through the adaptive tuning of controller parameters, the chosen objective function is the integral of time-weighted absolute error (ITAE) criterion [32], which is expressed as follows:

$$J = \int_0^\infty t|e|\,dt \quad (76)$$

The optimization problem can be represented as the following:

Minimize *J* under the constraint of:

$$K_p^{min} \le K_p \le K_p^{max} \quad (77)$$

$$K_d^{min} \le K_d \le K_d^{max} \quad (78)$$

In the optimization process, after completing a tour, each ant revises the pheromones left on the paths it traversed and adjusts the rules according to the following:

$$\tau(k)_{ij} = \tau(k-1)_{ij} + \frac{0.01\theta}{J} \quad (79)$$

Here, $\tau(k)_{ij}$ represents the pheromone value between nests *i* and *j* at the *k^th^* iteration, θ is the general coefficient governing pheromone updating, and J denotes the cost function for the tour undertaken by the ant. The pheromones on the path associated with both the best and worst tours of the ant colony undergo updating according to the following procedure:

$$\tau(k)_{ij}^{worst} = \tau(k)_{ij}^{worst} - \frac{0.3\theta}{J_{worst}} \quad (80)$$

$$\tau(k)_{ij}^{best} = \tau(k)_{ij}^{best} + \frac{\theta}{J_{best}} \quad (81)$$

In this context, τ^{best} and τ^{worst} represent the pheromones on the paths taken by the ant during the tour with the lowest cost value J_{best} and the tour with the highest cost value J_{worst} in a single iteration, respectively. Following pheromone evaporation, the ant algorithm discards its historical information and shifts its focus towards exploring new directions, avoiding entrapment in local minima, as follows:

$$\tau(k)_{ij} = 0\tau(k)_{ij}^{\lambda} + [\tau(k)_{ij}^{best} + \tau(k)_{ij}^{worst}] \quad (82)$$

Here, λ denotes the evaporation constant.

4.2 PROFINET based network integration

PROFINET, which stands for Process Field Network, is an industrial Ethernet standard used in automation and control systems. It is designed to facilitate real-time communication and data exchange in industrial environments, providing a foundation for the implementation of NCS. In our case study where a high-DOF robot manipulator is the controlled plant which has demanding performance requirements, the choosing of PROFINET relays mainly on the fact that is offers high-speed communication, enabling efficient data exchange. Additionally, PROFINET's network protocol ensures deterministic behavior, meaning that communication delays are predictable and can be tightly controlled [33]. Table 1 summarizes the protocols features for the chosen network.

Control network implementation transforms the structure of robot manipulator control, expands its applicability, and facilitates the realization of distributed control. Figure 6 illustrates a standard setup of a control system utilizing PROFINET.

The presented setup comprises two components: a network closed-loop system and a local closed-loop system. The controller initiates bus communication by transmitting the position reference to the actuators. Subsequently, the actuators employ this position reference to implement closed-loop control of the manipulator motors positions. Simultaneously, sensors periodically provide feedback data to the controller which then utilizes it to calculate a new position, initiating a cyclical repetition of the process. It is then evident that the number of sensors, actuators and motors is proportional to the number of DOF.

Table 1. PROFINET Protocols features

Features	PROFINET
RT Class	3
Clock Sync	Yes
Timeliness Approach	TDMA
Cycle time	31,25μs
Jitter	< 1μs
Determinism	Cycles
Topology	Any

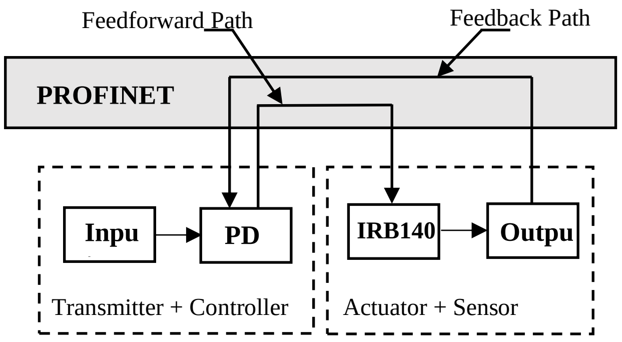


Figure 6. Control system implementation with PROFINET

4.3 Time delay consideration

Transmission of data through industrial networks introduces time delays in control loops due to factors such as network topology, traffic, and distance. In a PROFINET network, minimizing the NCS time delay is essential for achieving faster and more responsive communication among devices. However, in the NCS implementation, there are two types of time delay terms within the network. The first is a feedback time delay term $delay_{s-c}$ associated with the data transmission path from sensor to controller. The second is a feedforward time delay term $delay_{c-a}$ related to the data transmission path from controller to actuator. Figure 7 illustrates these two network delays.

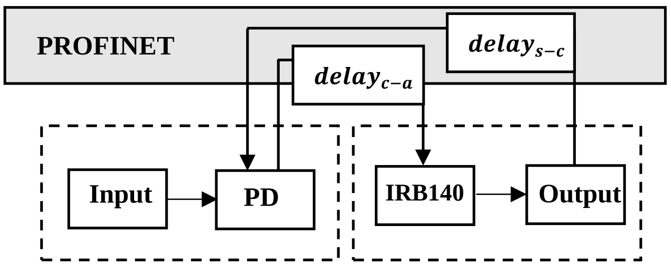


Figure 7. NCS-induced delay

Any delay in the controller can be assimilated into either of the two terms without compromising generality. A practical perspective acknowledges that these two transmission delays fluctuate for each data transfer. Consequently, it is anticipated that a controller design approach with adaptive capabilities will help in adjusting to variable time delays. Nevertheless, this alone is generally insufficient for ensuring the preservation of the desired control system performance.

4.3.1 Network delays on S-C path

During each sampling period T_s, sensor data are emitted after being processed by sensor task J_s , the data wait acquisition by the S-C network task J_{net_sc}. The start time of sensor task *n* in the k_{th} control cycle is determined as follows:

$$S_s^n(k) = S_s^n(0) + kT_s \tag{83}$$

With global clock synchronization, sensors can be configured such that their start times are identical. Knowing that the *sync* signal cycle can align with the bus cycle, an input shift delay is introduced after the sync event, and inbound data are secured once the delay has expired. This delay must be set to ensure that sufficient time for sensor data processing is reserved, allowing the sensor data to be available prior to the arrival of the S-C frames which evidently will minimize the waiting time for sensor data [34].

Subsequently, sensor data are saved in the network slave controllers but are not transmitted until the following bus cycle if clock synchronization is not used. This raises the potential for an extra waiting delay $T_w^{J_s^n \to J_{net\_sc}}(k)$, which is between 0 and T_s. Clock synchronization results in dependable and predictable system behavior. In order to reduce the latency between the sensor task J_s and J_{net_sc}, clock synchronization would be ideal. Taking into account both situations, the bus cycle's start time can be expressed as follows [35]:

$$S_{net\_sc}(k) = S_s^n(k) + D_s + (1 - \lambda_{sync})\, T_w^{J_s^n \to J_{net\_sc}}(k) \tag{84}$$

where, λ_{sync} is a boolean parameter that indicates the use of clock synchronization, and D_s is the sensor execution time. When clock synchronization is used, $\lambda_{sync} = 1$. In both situations, the cyclic communication structure eliminates the need for a local queue on sensors.

All sensor data is gathered in the network master controller at the end of a communication cycle. In order to read the sensor input data and begin calculating actuator set-points using the PD algorithm, the network master controller sends an interrupt to the controller. The controller start time is expressed as follows:

$$S_c(k) = S_{net\_sc}(k) + D^n_{net\_sc} + T_w^{J_{net\_sc} \to J_c}(k) \qquad (85)$$

where, $T_w^{J_{net\_sc} \to J_c}$ is the time required by the controller to retrieve the data, and $D^n_{net\_sc}$ is the network delay for the data collected at sensor n to reach the controller.

The final S–C network delay can be determined by measuring the time interval between the end of the sensor task J^n_s and the beginning of the controller task J_c, i.e.

$$\begin{aligned} delay^n_{sc,network}(k) &= S_c(k) - (S^n_s(k) + D_s) \\ &= D^n_{net\_sc} + (1 - \lambda_{sync}) T_w^{J^n_s \to J_{net\_sc}}(k) \\ &\quad + T_w^{J_{net\_sc} \to J_c}(k) \end{aligned} \qquad (86)$$

4.3.2 Network delays on C-A path

The *C-S* network task J_{net_ca} can begin once the controller completes its computation and the outgoing process data have been transferred to the network.

$$S_{net\_ca}(k) = S_c(k) + P_c + T_w^{J_c \to J_{net\_ca}}(k) \qquad (87)$$

where, P_c the actuation deadline task. Upon receiving the output data from the network via J_{e_ca}, the task of actuator n is initiated:

$$S^n_a(k) = S_{net\_ca}(k) + D^n_{net\_ca} \qquad (88)$$

where, $D^n_{net\_ca}$ is the communication delay between the beginning of the bus cycle and the actuator n receiving the output data. After a bus cycle is initiated, the actuators typically get data over the network in less than a full bus cycle. The node index n determines how each actuator endure $D^n_{net\_ca}$. In contrast, $D^n_{net\_sc}$ is the same for every sensor in the case of clock-driven S-C delay.

The actuators begin execution as soon as their network slave controllers receive data, with actuation occurring within D_a (actuator execution time). By disregarding the actuators data access time, the C-A network delay can be calculated as:

$$\begin{aligned} delay^n_{ca,network}(k) &= S^n_a(k) - (S_c(k) + P_c) \\ &= D^n_{net\_ca} + T_w^{J_c \to J_{net\_ca}}(k) \end{aligned} \qquad (89)$$

5. SIMULATIONS AND RESULTS

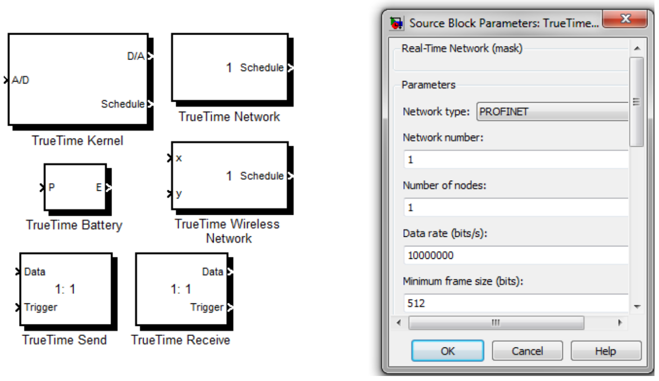
This section aims to conduct system simulations using the selected controller to assess asymptotic stability through network integration. Successful completion of this task will validate the mathematical proof detailed in section 4.1 for the model. The inclusion of gravity compensation transforms the model as expressed in Eq. (69), resulting in the following system input:

$$u = -K_p \tilde{q} - K_d \dot{q} \qquad (90)$$

The mathematical proof provides no additional constraints on K_p and K_d aside from requiring them to be positive definite. Optimizing these controller gains will play a critical role in achieving global asymptotic stability.

A MATLAB toolbox called TrueTime has been developed specifically for simulating the communication and control dynamics of network-based real-time control systems [36]. The toolbox is often used to model and analyze the performance of control systems in the presence of communication delays and network uncertainties, and it currently supports a fair number of network protocols including PROFINET.

The tool includes not only a library but also a suite of functions, available in either C++ files or Matlab Mex files. These functions are specifically made to configure and delineate the execution of tasks within the kernel block. They include various functionalities such as AD/DA conversion, message transmission and reception, task creation and management, handling interrupts, managing timers, and monitoring events as represented in Figure 8. The TrueTime tool seamlessly integrates with Simulink libraries, enabling direct connection to existing blocks. Consequently, the entire networked control system can be effectively modeled within the Simulink environment.



Real-Time Network (mask)

Parameters

Network type: PROFINET

Network number: 1

Number of nodes: 1

Data rate (bits/s): 10000000

Minimum frame size (bits): 512

Figure 8. PROFINET network block in TrueTime

Three simulations are discussed, each featuring distinct initial conditions and reference values to illustrate the system's behavior. Initial parameters and reference values are established as follows:

Simulation 1:

$q_{init}=[0 \; \frac{\pi}{2} \; -\frac{\pi}{2} \quad 0 \quad 0 \quad 0\,]^T$

$\dot{q}_{init}=[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\,]^T$

$\dot{q}_{ref}=[-\pi \; \pi \; -\pi \; -\pi \; -\frac{\pi}{2} \; \pi\,]^T$

Simulation 2:

$q_{init}=[0 \quad \frac{\pi}{2} \quad -\frac{\pi}{2} \quad 0 \quad 0 \quad 0\,]^T$

$\dot{q}_{init}=[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\,]^T$

$\dot{q}_{ref}=[-\pi \quad \pi \quad -\pi \quad -\pi \quad -\frac{\pi}{2} \quad \pi\,]^T$

Simulation 3:

$q_{init}=[0 \quad \frac{\pi}{2} \quad -\frac{\pi}{2} \quad 0 \quad 0 \quad 0\,]^T$

$\dot{q}_{init}=[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\,]^T$

$\dot{q}_{ref}=[0 \quad -\pi \quad \pi \quad \frac{\pi}{2} \quad -\pi \quad 0\,]^T$

Table 2 shows the ACO parameters used in the simulation process.

Table 2. ACO parameters

Parameters	Value
Population size	8
Evaporation constant λ	0.5
Pheromone initial value τ	100
Maximum number of iterations	100

Joint variables responses followed by the control input for each simulation are illustrated in Figures 9-14 respectively.

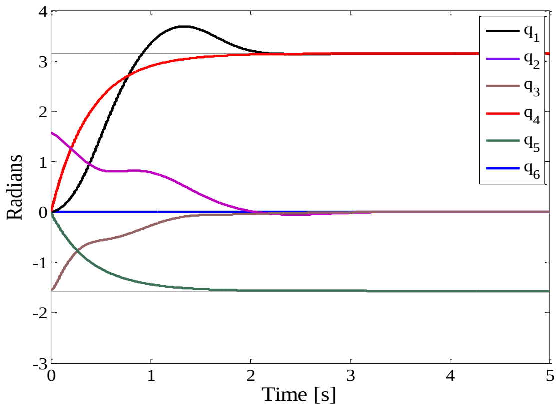


Figure 9. Networked feedback position control, simulation 1

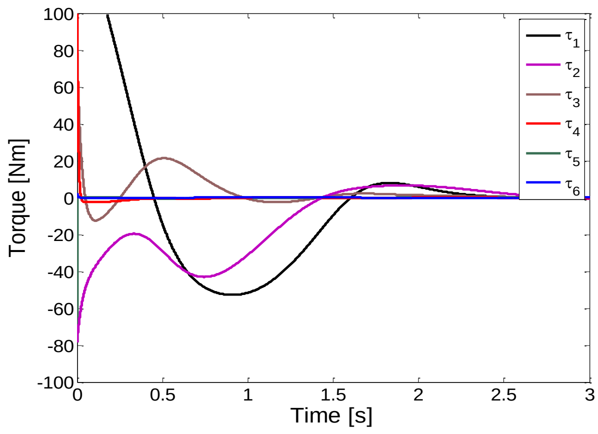


Figure 10. Networked control input, simulation 1

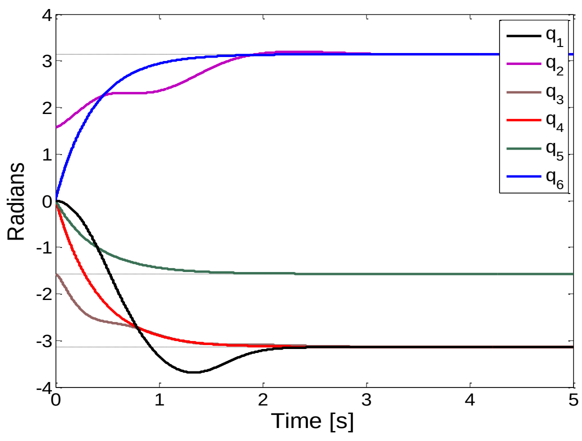


Figure 11. Networked feedback position control, simulation 2

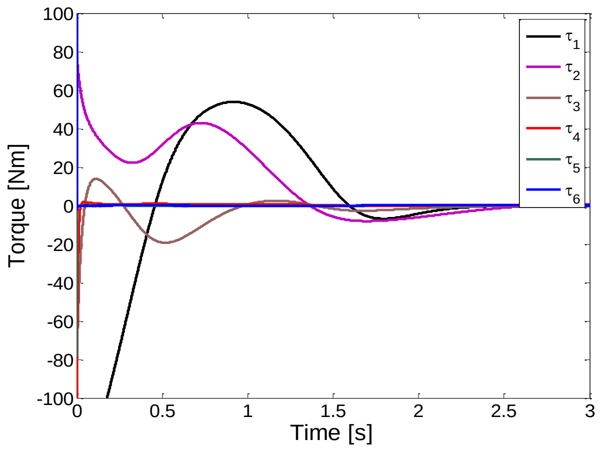


Figure 12. Networked control input, simulation 2

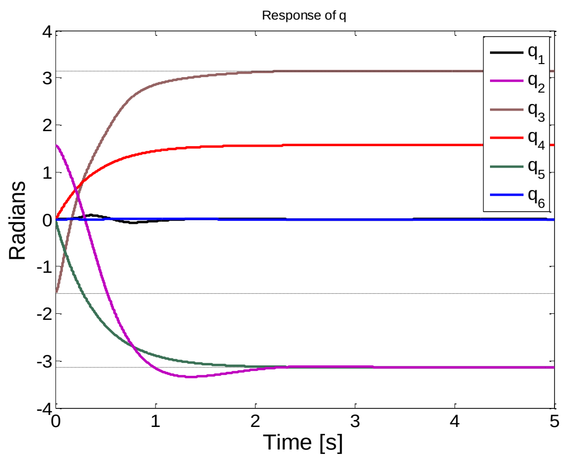


Figure 13. Networked feedback position control, simulation 3

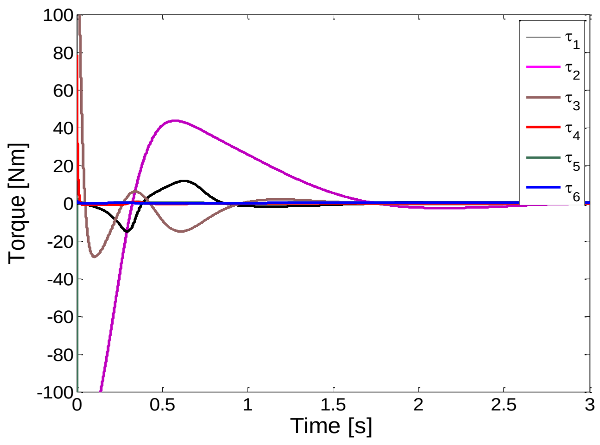


Figure 14. Networked control input, simulation 3

The resulting optimal PD coefficients for this simulation are as follows:

$$K_p = \begin{bmatrix} 50.05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4.89 & 0 & 0 & 0 & 0 \\ 0 & 0 & 25.4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50.11 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.04 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 19.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20.03 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.02 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.012 \end{bmatrix}$$

The resulting optimal PD coefficients for this simulation are as follows:

$$K_p = \begin{bmatrix} 38.51 & 0 & 0 & 0 & 0 & 0 \\ 0 & 14.22 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4.21 & 0 & 0 & 0 \\ 0 & 0 & 0 & 41.87 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 48.06 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 14.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.44 & 0 & 0 & 0 \\ 0 & 0 & 0 & 14.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.07 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16.03 \end{bmatrix}$$

The resulting optimal PD coefficients for this simulation are as follows:

$$K_p = \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 44.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.99 & 0 & 0 \\ 0 & 0 & 0 & 0 & 15.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.33 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 1.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 24.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.87 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.52 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

The execution of tasks in the forward and backward paths over PROFINET network can be studied in detail in Figure 15 and Figure 16.

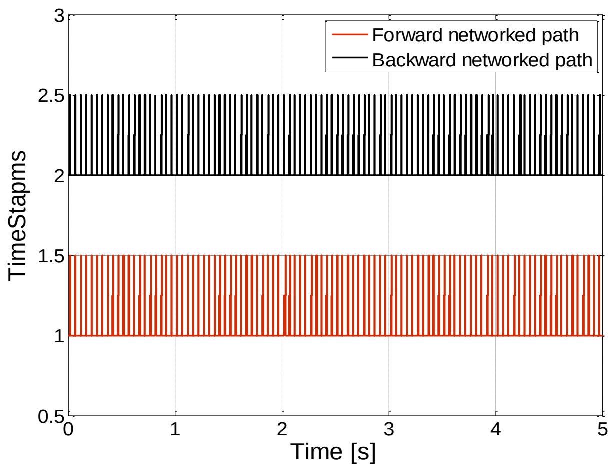


Figure 15. PROFINET network schedules

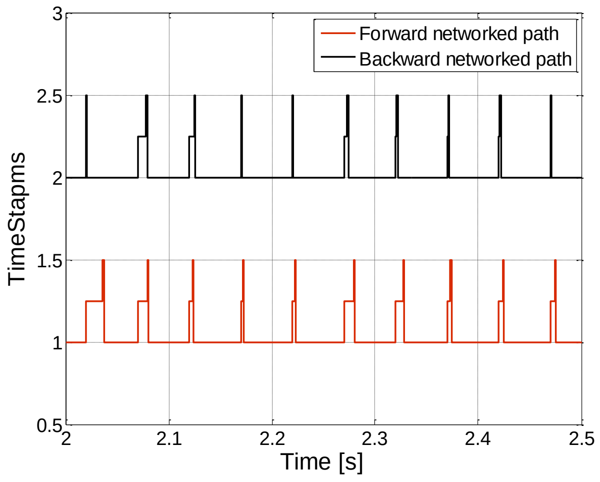


Figure 16. Closeup of PROFINET network schedules

5.1 Analysis and comments

Each simulation indicates that the system's states consistently converge to the desired reference within approximately 3 seconds, showcasing rapid response dynamics. The position responses of most joints exhibit negligible overshoot, which is a crucial indicator of robust performance, especially in high-DOF robotic manipulators. This minimal overshoot is particularly advantageous in industrial applications where precision and stability are critical to ensure the safety and efficiency of robotic operations. Moreover, the results highlight the effectiveness of the proposed networked control framework. The ITAE index closely matches the values typically observed in traditional local control schemes. This outcome confirms that the incorporation of networked control, even with the presence of network-induced delays, does not degrade the system's ability to achieve accurate and reliable position tracking. Instead, it underscores the proposed adaptive robust PD control law in mitigating the effects of delays and maintaining high performance. The verification of asymptotic stability through Lyapunov analysis further strengthens the reliability of the approach. The simulated system consistently demonstrates globally stable behavior, with all joint positions settling at their references without divergence or oscillation. This stability, combined with the high accuracy of the position tracking, validates the robustness of the proposed methodology.

However, it is important to underscore several crucial elements. Firstly, it should be noted that actuators have limitations in providing infinite torque. The maximum input torque is restricted by the nominal torque of the actuators and their gear ratio. While this constraint can be easily incorporated into Simulink by saturating the input, it is not deemed necessary for the following reasons. In Section 4.1, gravitational terms were introduced to the input under the assumption of constant and known gravitational acceleration. This presupposes that the maximum input torque in the actuators surpasses *g(q)*. Although the data sheets for the manipulator do not specify torque values or any additional attributes, this assumption remains valid, as empirical observations demonstrate the robot effectively overcome gravity in a real-world setting. Given that there are no additional constraints on the input torque, this confirms global asymptotic stability even in the presence of saturated inputs.

The only notable difference in the simulations lies in the prolonged time required to attain a steady state.

Furthermore, actuators are incapable of instantaneously altering the input torque value. In simpler terms, the input cannot exhibit a perfect step reference. The manipulator relies on AC-motors for control, delivering torque through a magnetic field. The speed at which this magnetic field is established defines the peak rate of fluctuation in input torque. While fluctuation limiters can be incorporated into Simulink, it is assumed that AC-motors rapidly establish their electric fields, making fluctuation limiters irrelevant in the simulations.

5.2 Limitations

Certain limitations have been intentionally selected. Initially, joint friction is ignored for two primary factors. Firstly, estimating friction parameters without provided information would be akin to a random guess. Secondly, its inclusion does not significantly impact simulations when the input is not saturated. Nevertheless, if joint friction were to be considered, the most straightforward approach would be to exclusively model linear friction. The resulting model would then be as follows:

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+F_v\dot{q} + \dot{q}+g(q)=u \tag{91}$$

where, F_v represent the diagonal matrix of joint friction coefficients.

It is worth noting that various control strategies and approaches can be employed and integrated in a well-chosen network setting. The selection of the appropriate control methodology should align with the specific operational needs of the manipulator. For instance, certain tasks may demand precise movement within specific time intervals. Factors such as the manipulator's design, motor specifications, and issues related to backlash and friction can also influence the decision on the appropriate control technique.

Finally, the proposed methodology outlined in the preceding sections can be implemented in practical settings by integrating improvements in hardware, exploiting established industry theories and facts. This may involve actions like increasing network communication speed, segmenting the PROFINET network, and minimizing transmission delay through a store-and-forward approach in relevant network components to achieve enhanced control effects.

6. CONCLUSION

This paper outlines a methodology for networked control of high-DOF robot manipulator demonstrated through a detailed case study of the IRB140. Initially, it has been demonstrated that achieving accurate estimation of dynamic parameters poses a difficult and time-intensive task. This necessitates either the ability to measure the state variables during manipulator motion or specialized knowledge of alternative identification techniques. In the case of the IRB 140 model, parameters were estimated through meticulous examination of the manipulator, occasionally relying on intuitive conjecture when necessary. Additionally, the integration of the network within the closed-loop control system of the manipulator has been demonstrated in successive stages, explaining the crucial considerations for selecting the right network protocol and mitigating the adverse effects of the network on the feedback control system. The analysis highlights PROFINET as a proficient network option for NCS applications, particularly beneficial for highly dynamic manipulators with intricate models.

The results highlight the efficacy of the proposed networked control methodology in achieving precise and stable position tracking for high-DOF robot manipulators, even in the presence of network-induced delays. This demonstrates a significant advancement in bridging the gap between traditional local control methods and modern networked control requirements in industrial automation. The ability of the proposed framework to closely match the performance indices (e.g., ITAE) of local control systems suggests that networked control can be implemented without sacrificing precision or stability. This is particularly impactful for high-DOF manipulators operating in distributed industrial environments where networked control is becoming increasingly necessary for scalability, remote monitoring, and integration with industry technologies. Furthermore, the proven asymptotic stability and robust response under various conditions lay the groundwork for the reliable deployment of such systems in dynamic and uncertain environments.

REFERENCES

[1] Mittapally, H., Ghosh, S., Kamal, S. (2023). Predictive control of networked control system with event-triggering in two channels. European Journal of Control, 72: 100810. https://doi.org/10.1016/j.ejcon.2023.100810

[2] Wang, H., Wang, J., Xu, H., Zhao, S. (2023). A self-triggered stochastic model predictive control for uncertain networked control system. International Journal of Control, 96(8): 2113-2123. https://doi.org/10.1080/00207179.2022.2084163

[3] El Abbadi, R., Jamouli, H. (2023). Fault detection of a networked control system and its application to a DC motor. International Journal of Control, Automation and Systems, 21(6): 1769-1779. https://doi.org/10.1007/s12555-022-0339-6

[4] Sakthivel, R., Santra, S., Mathiyalagan, K., Su, H. (2015). Robust reliable control design for networked control system with sampling communication. International Journal of Control, 88(12): 2510-2522. https://doi.org/10.1080/00207179.2015.1048294

[5] Dahake, V.R., Patil, M.D., Vyawahare, V.A. (2024). Analysis of networked control system with integer-order and fractional-order PID controllers. International Journal of Control, Automation and Systems, 22(2): 373-386. https://doi.org/10.1007/s12555-023-0181-5

[6] Rahmani, B., Belkheiri, M. (2019). Adaptive neural network output feedback control for flexible multi-link robotic manipulators. International Journal of Control, 92(10): 2324-2338. https://doi.org/10.1080/00207179.2018.1436774

[7] Xia, J., Zhang, Y., Yang, C., Wang, M., Annamalai, A. (2019). An improved adaptive online neural control for robot manipulator systems using integral Barrier Lyapunov functions. International Journal of Systems Science, 50(3): 638-651. https://doi.org/10.1080/00207721.2019.1567863

[8] Zhao, X., Liu, Z., Zhu, Q. (2023). Neural network-based adaptive controller design for robotic manipulator subject to varying loads and unknown dead-zone.

Neurocomputing, 546: 126293. https://doi.org/10.1016/j.neucom.2023.126293

[9] Garcia-Hernandez, R., Lopez-Franco, M., Sanchez, E.N., Alanis, A.Y., Ruz-Hernandez, J.A. (2017). Decentralized Neural Control: Application to Robotics. Springer, Cham, Switzerland.

[10] Goharimanesh, M., Mehrkish, A., Janabi-Sharifi, F. (2020). A fuzzy reinforcement learning approach for continuum robot control. Journal of Intelligent & Robotic Systems, 100(3): 809-826. https://doi.org/10.1007/s10846-020-01237-6

[11] Cronin, J., Escano, J.M., Roshany-Yamchi, S., Canty, N. (2014). Fuzzy-based generalized predictive control of a robotic arm. In 25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014), Limerick, IET, pp. 153-157. https://doi.org/10.1049/cp.2014.0676

[12] Tang, L., Liu, Y.J., Tong, S. (2014). Adaptive neural control using reinforcement learning for a class of robot manipulator. Neural Computing and Applications, 25: 135-141. https://doi.org/10.1007/s00521-013-1455-2

[13] Purwar, S., Kar, I.N., Jha, A.N. (2008). Adaptive output feedback tracking control of robot manipulators using position measurements only. Expert Systems with Applications, 34(4): 2789-2798. https://doi.org/10.1016/j.eswa.2007.05.030

[14] Ullah, H., Malik, F.M., Raza, A., Mazhar, N., Khan, R., Saeed, A., Ahmad, I. (2021). Robust output feedback control of single-link flexible-joint robot manipulator with matched disturbances using high gain observer. Sensors, 21(9): 3252. https://doi.org/10.3390/s21093252

[15] Ghoul, A., Kara, K., Benrabah, M., Hadjili, M.L. (2022). Optimized nonlinear sliding mode control of a continuum robot manipulator. Journal of Control, Automation and Electrical Systems, 33(5): 1355-1363. https://doi.org/10.1007/s40313-022-00914-1

[16] Jung, S. (2023). Sliding mode control for a hybrid force control scheme of a robot manipulator under uncertain dynamics. International Journal of Control, Automation and Systems, 21(5): 1634-1643. https://doi.org/10.1007/s12555-022-0756-6

[17] Rigatos, G., Siano, P., Raffo, G. (2016). An H-infinity nonlinear control approach for multi-DOF robotic manipulators. IFAC-PapersOnLine, 49(12): 1406-1411. https://doi.org/10.1016/j.ifacol.2016.07.766

[18] Capisani, L.M., Facchinetti, T., Ferrara, A. (2010). Real-time networked control of an industrial robot manipulator via discrete-time second-order sliding modes. International Journal of Control, 83(8): 1595-1611. https://doi.org/10.1080/00207179.2010.48407

[19] Nath, K., Bera, M.K. (2023). Integral sliding mode control of networked robotic manipulator: A dynamic event-triggered design. Advanced Robotics, 37(1-2): 141-153. https://doi.org/10.1080/01691864.2022.2123255

[20] Kheirkhah, A., Aschenbrenner, D., Fritscher, M., Sittner, F., Schilling, K. (2015). Networked control systems with application in the industrial tele-robotics. IFAC-PapersOnLine, 48(10): 147-152. https://doi.org/10.1016/j.ifacol.2015.08.123

[21] Nanu, L., Colangelo, L., Novara, C., Montenegro, C.P. (2024). Embedded model control of networked control systems: An experimental robotic application. Mechatronics, 99: 103160. https://doi.org/10.1016/j.mechatronics.2024.103160

[22] Ghasemzadeh Ebli, H., Nekoui, M.A. (2019). Multirate control of nonlinear robotic manipulators with network structure under bounded disturbances. Journal of Vibration and Control, 25(18): 2523-2533. https://doi.org/10.1177/1077546319858847

[23] ABB Library. IRB 140 Data Sheet. https://library.abb.com/d/PR10031EN_R15_HR.

[24] ABB Library. IRB 140 Product Specication. https://library.abb.com/d/3HAC041346-001.

[25] Bajd, T., Mihelj, M., Lenarcic, J., Stanovnikb, A., Munih, M., Rejc, J., Slajpah, S. (2010). Robotics. Springer. https://www.springerprofessional.de/robotics/15978442#TOC.

[26] Beatty, M. (2006). Principles of Engineering Mechanics. Volume 2 Dynamics: The Analysis of Motion. Springer Verlag. https://doi.org/10.1007/978-0-387-31255-2

[27] Siciliano, B., Khatib, O., Kroger, T. (2016). Springer Handbook of Robotics. Springer. https://doi.org/10.1007/978-3-319-32552-1

[28] Wei, B., Zhang, D. (2021). A review of dynamic balancing for robotic mechanisms. Robotica, 39(1): 55-71. https://doi.org/10.1017/S0263574720000168

[29] Sciavicco, L., Siciliano, B. (2012). Modelling and control of robot manipulators. Springer Science & Business Media.

[30] Spong, M.W., Hutchinson, S., Vidyasagar, M. (2006). Robot Modeling and Control. Wiley, New Jersey.

[31] Dorigo, M., Stutzle, T. (2004). Ant Colony Optimization. MIT Press. https://doi.org/10.7551/mitpress/1290.001.0001

[32] Martins, G. (2005). Tuning PID controllers using ITAE criterion. International Journal of Engineering, 21(5): 867-873.

[33] Wisniewski, L. (2017). New Methods to Engineer and Seamlessly Reconfigure Time Triggered Ethernet Based Systems During Runtime Based on the PROFINET IRT Example. Springer Vieweg. https://doi.org/10.1007/978-3-662-54650-5

[34] Wu, X., Xie, L., Lim, F. (2014). Network delay analysis of EtherCAT and PROFINET IRT protocols. In IECON 2014-40th Annual Conference of the IEEE Industrial Electronics Society, Dallas, TX, USA, pp. 2597-2603. https://doi.org/10.1109/IECON.2014.7048872

[35] Robert, J., Georges, J.P., Rondeau, É., Divoux, T. (2012). Minimum cycle time analysis of Ethernet-based real-time protocols. International Journal of Computers, Communications and Control, 7(4): 743-757.

[36] Ohlin, M., Henriksson, D., Cervin, A. (2010). TrueTime 2.0 reference manual. Internal Report, Department of Automatic Control, Lund University, Sweden. http://archive.control.lth.se/media/Research/Tools/TrueTime/report_2016-02-10.pdf.