An Integrated Approach to Robotic Joint Interpolation: Kinematic Modeling and Constraints for Smooth Trajectories

Luis Antonio Orbegoso Moreno, Edgar David Valverde Ramírez, José Luis Ruíz Rodríguez, Isac Daniel Miñano Corro
National University of Trujillo, Trujillo, Peru
lorbegoso@unitru.edu.pe

Abstract — **This paper introduces a direct trajectory planning approach for manipulator robots, seamlessly transitioning from the task space to the robot joint space, while adhering to kinematic (joint position and velocity boundaries) and dynamic (torque limits) constraints. The proposed method employs state-space modeling utilizing joint positions and their derivatives up to order n-1 as state variables, and the nth derivative of joint positions as the control signal to guarantee continuity and differentiability during interpolation. Furthermore, the output of the state-space model created are the forward and differential kinematics of the robot, enabling interpolation with consideration for posture (position and orientation) and velocity (linear and angular) in the task space. Thereby, the optimal control signal computation is derived via the minimization of the Square Error between the state-space's output with the desired final output through quasi-Newton optimization. Then, the calculation of joint values over time is achieved via the nth integration of the previously computed control signal using the Cauchy formula for repeated integration. Finally, the proposed method is validated on the ABB-IRB-120 robot model, where it demonstrates a unified solution for task-space path planning under both kinematic and dynamic constraints.**

Keywords - robotic trajectories, kinematic interpolation, robotic manipulator, inverse kinematics

I. INTRODUCTION

In manipulator robot path planning, the conventional approach typically involves a sequential process. It begins with deriving inverse kinematics (IK) for each point in the task space. Subsequently, interpolation is performed at the respective time interval for each joint value obtained from the IK solution [1].

Firstly, IK involves determining the joint values necessary for a kinematic chain to reach a specified point in its workspace being able to coincide both in position and orientation [2]. Conventional approaches to solving this problem include analytical solvers and numerical techniques like Newton-Raphson or Jacobian inverse [3]. However, other strides in metaheuristic methods have introduced innovative stochastic solutions to the inverse kinematics puzzle among which stand out Particle Swarm Optimization (PSO) [4, 5] and Genetic Algorithms [6, 7]. With respect to heuristic methods, FABRIK is an algorithm introduced by [8] for computer graphics which outstands by its fast IK solution's convergence just using vector operations; reason why FABRIK has had adaptations in the field of robotics [9, 10, 11].

Continuing, interpolation-based path planning methods involve determining joint values that connect other joint values within a specified time interval and satisfying defined boundary conditions [12]. In [13], a trajectory planning approach for a 6-degree-of-freedom (DoF) modular manipulator is proposed, utilizing polynomial interpolation, demonstrating enhanced trajectory accuracy and efficiency compared to alternative methods. Similarly, [14] introduces a reverse splitting and local interpolation method to optimize the motion trajectory of dual manipulator robots, resulting in the removal of redundant path nodes and improved path quality. Also, in [15] a bidirectional interpolation method for sampling-based path planning algorithms is presented, reducing path lengths, and enhancing path shape compared to other algorithms. For smooth interpolations in the task space, [16] employs dual quaternions interpolation using the SLERP technique, yet this approach, while accommodating velocity conditions, is limited to creating speed curves of class C1, indicating non-differentiability in acceleration within its interpolation domain. To address this limitation, [17] introduces interpolation based on logarithmic quaternion generating curves, offering differentiability as desired; however, this method does not consider speed conditions.

On the other hand, the derivatives of angular positions in robots must be differentiable and continuous [18] to generate smooth angular trajectories without singularities. This ensures that joint torques, directly proportional to acceleration and the square of joint speed according to Lagrange-Euler dynamic modeling [19], remain continuous and differentiable, ensuring the robot actuators' smooth and precise movement [18]. This requirement is crucial for optimal end-effector movement, as emphasized in [20] with parallel Dexterous Twin Arms Robots (DexTAR).

The present work presents a streamlined trajectory planning methodology, deviating from the conventional approach by initiating with state-space modeling of robot kinematics in joint space, which incorporates joint positions and their derivatives up to order n-1 as state-space variables, and the control signal defined as the nth derivative of joint positions, to ensure continuity and differentiability during interpolation. For the output of the state-space model, forward and differential kinematics of the robot are considered, facilitating trajectory interpolation using the task space information as the target to be reached through the control signal computed using quasi-Newton optimization, which also incorporates kinematic and dynamic constraints—joint positions, velocities, and torque limits— to guarantee a cohesive trajectory. Thereby, once the control signal is computed, its nth integration is performed via the Cauchy formula which results in the joint values over time (interpolation). These results are reflected in the tests on the ABB-IRB-120 robot model, which validates the effectiveness of the proposed method in unified path planning within the task space under kinematic and dynamic constraints.

The article structure is outlined as follows: Section II presents the robotic model (kinematics and dynamics parameters), state-space modeling of robot joint's kinematics, and the optimization for finding the control signal with subsequent integration for interpolation. In Section III, results of the interpolation in the task space and in the joint space are detailed, where for the first and fourth joints of the robot are illustrated the angular positions with their derivatives and torques, also with their respective boundaries. Finally, section IV gives the work's final conclusions.

II. MATERIALS AND METHODS

A. The Robot Model

The robotic model used in this work was the ABB-IRB-120, which is a 6-DoF compact (25 kg mass), and precise industrial robot manufactured by ABB, recognized for its small size and high payload capacity [21]. Fig 1 shows the robotic model, indicating the inertial reference frame, the kinematic parameters, its respective enumerated joints, and its links highlighted with different colors starting from the purple one as noted in the legend.

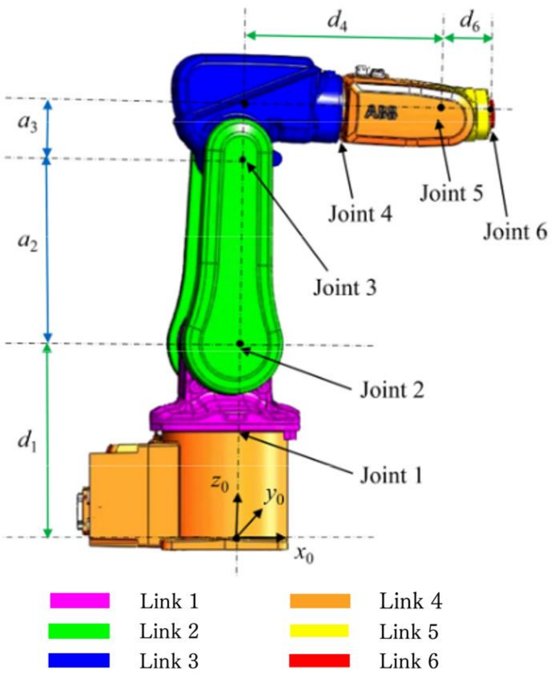

Figure 1. Links and joints of the ABB-IRB-120 robot [21]

The Denavit-Hartenberg (DH) parameters of the robotic manipulator that complement the previous image are shown in the following table.

TABLE I. DENAVIT-HARTENBERG PARAMETERS OF THE ABB-IRB-120 ROBOT

N° joint	Parameters			
	θ_i (rad)	d_i (mm)	α_i (rad)	a_i (mm)
1	θ_1	290	-½π	0
2	θ_2	0	0	270
3	θ_3	0	½π	70
4	θ_4	374	-½π	0
5	θ_5	0	½π	0
6	θ_6	75	0	0

Likewise, the data on the maximum and minimum joint values, the maximum velocities and maximum torques that each of the joints of the ABB-IRB-120 robotic manipulator can reach are found in Table II.

TABLE II. ABB-IRB-120 ROBOT JOINT LIMITS

N° joint	Joint position (deg°)		Max joint velocity (deg°/s)	Max joint torque (N.m)
	min	*max*		
1	-165	165	250	4.8
2	-110	110	250	4.8
3	-70	110	250	4.8
4	-160	160	320	4.8
5	-120	120	320	4.8
6	-400	400	420	2.2

In the same way, in Table III are shown the dynamic parameters of the robotic model, where the center of mass is expressed relative to the frame of each link and the inertia tensors are expressed with respect to the center of mass of each link.

TABLE III. DYNAMIC PARAMETERS OF THE ABB-IRB-120 ROBOT

N° link	Center of mass (m)			Inertia Tensor (kg.m^2)	Mass (kg)
	x	*y*	*z*		
1	-0.05	0.0	0.01	$\begin{bmatrix} 0.012 & 0 & 0 \\ 0 & 0.011 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$	0.29
2	0.0	0.0	0.124	$\begin{bmatrix} 0.064 & 0 & 0 \\ 0 & 0.012 & 0 \\ 0 & 0 & 0.064 \end{bmatrix}$	0.27
3	0.0	0.024	0.058	$\begin{bmatrix} 0.35 & -0.23 & 0 \\ -0.23 & 0.18 & 0 \\ 0 & 0 & 0.37 \end{bmatrix}$	0.07
4	-0.09	0.0	0.0	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.008 & 0 \\ 0 & 0 & 0.006 \end{bmatrix}$	0.347
5	0.0	0.06	0.0	$\begin{bmatrix} 0.002 & 0 & 0 \\ 0 & 0.002 & 0 \\ 0 & 0 & 0.002 \end{bmatrix}$	0.07
6	-0.09	0.0	0.0	$\begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}$	0.002

B. State-Space Kinematic Modeling

One of the simplest motions studied in kinematics is the Uniformly Accelerated Rectilinear Motion (UARM), which exists under the assumption that acceleration is constant over a time interval Δt [22]. However, if instead of the acceleration being constant in the given time interval Δt, more generally, the nth derivative of position $\frac{d^n x}{dt^n}$ is instead. So, the equation system that describes the next position x_{i+1} and its derivatives after Δt is shown below.

$$\begin{gathered} x_{(i+1)} = x_{(i)} + \frac{dx_{(i)}}{dt}\Delta t + \frac{d^2x_{(i)}}{dt^2}\frac{\Delta t^2}{2} + \dots + \frac{d^n x_{(i)}}{dt^n}\frac{\Delta t^n}{n!} \\ \frac{dx_{(i+1)}}{dt} = \frac{dx_{(i)}}{dt} + \frac{d^2x_{(i)}}{dt^2}\Delta t + \dots + \frac{d^n x_{(i)}}{dt^n}\frac{\Delta t^{n-1}}{(n-1)!} \\ \vdots \\ \frac{d^{n-1}x_{(i+1)}}{dt^{n-1}} = \frac{d^{n-1}x_{(i)}}{dt^{n-1}} + \frac{d^n x_{(i)}}{dt^n}\Delta t \end{gathered} \tag{1}$$

This equation system can be rewritten using the discrete time state-space matrices notations as is shown below.

$$\vec{x}_{(i+1)} = A\vec{x}_{(i)} + Bu_{(i)} \tag{2}$$

Where $\vec{x}_{(i)} = \left[x_{(i)}, \frac{dx_{(i)}}{dt}, \dots, \frac{d^{n-1}x_{(i)}}{dt^{n-1}}\right]^T$ is a column vector that has the information of a single variable and its derivatives at the moment "i"; also $u_{(i)} = \left[\frac{d^n x_{(i)}}{dt^n}\right]$ is the control vector that in this case just has a single scalar value corresponding with the nth derivative of the position. Thus, the elements of the matrices A and B are shown in the equalities (3) and (4) respectively, which are just in function of Δt and the position derivative order "n".

$$A = \begin{bmatrix} 1 & \Delta t & \dots & \frac{\Delta t^{n-1}}{(n-1)!} \\ 0 & 1 & \dots & \frac{\Delta t^{n-2}}{(n-2)!} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}_{[n\times n]} \tag{3}$$

$$B = \begin{bmatrix} \frac{\Delta t^n}{n!} \\ \frac{\Delta t^{n-1}}{(n-1)!} \\ \vdots \\ \Delta t \end{bmatrix}_{[n\times 1]} \tag{4}$$

Nevertheless, the equation (2) could be used just to predict the values of a single variable (single joint) and its derivatives. So, to add all the "m" joints of a robot in a single state-space model, the equation (5) is presented, where the column vector $X_{(i)} = \left[\vec{x}_1^T, \vec{x}_2^T, \dots, \vec{x}_j^T, \dots, \vec{x}_m^T\right]_{(i)}^T$ owns orderly all the m joints of the robot and its derivatives at the moment "i"; also the same with the vector $U_{(i)} = \left[u_1, u_2, \dots, u_j, \dots, u_m\right]_{(i)}^T$, which has the nth derivative for all the robot joints and works as the control vector for the system.

$$X_{(i+1)} = \mathbb{A}X_{(i)} + \mathbb{B}U_{(i)} \tag{5}$$

In addition, the matrices $\mathbb{A}$ and $\mathbb{B}$ are created by repeatedly placing the matrices A and B diagonally respectively according with the robot's DoF "m" as is shown in equalities (6) and (7).

$$\mathbb{A} = \begin{bmatrix} A & 0 & \dots & 0 \\ 0 & A & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A \end{bmatrix}_{[n\cdot m\times n\cdot m]} \tag{6}$$

$$\mathbb{B} = \begin{bmatrix} B & 0 & \dots & 0 \\ 0 & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B \end{bmatrix}_{[n\cdot m\times m]} \tag{7}$$

Now, the output of the system is assumed in this work to be the forward kinematic (FK), computed with homogeneous transformation matrices, whose rows then are stacked with the velocity of the robot's end-effector as shown in equation (8). But it can be added other variables in function of the joint's acceleration, jerk, jounce, etc., according to the conditions of the trajectory to be interpolated. The FK is in function of the articulation positions which are represented by the articulation value vector $\vec{q}_{(i)} = \left[x_1, \dots, x_j, \dots, x_m\right]_{(i)}^T$; similar way with the end-effector velocity which is the result of the product of the Jacobian matrix of the FK with the articulation velocity vector $\vec{\dot{q}}_{(i)} = \left[\frac{dx_1}{dt}, \dots, \frac{dx_j}{dt}, \dots, \frac{dx_m}{dt}\right]_{(i)}^T$. Both $\vec{q}_i$ and $\vec{\dot{q}}_i$ can be formed by rearranging the state vector X_i.

$$Y_{(i)} = \begin{bmatrix} FK(\vec{q}_{(i)}) \\ J(\vec{q}_{(i)}) \cdot \vec{\dot{q}}_{(i)} \end{bmatrix} \tag{8}$$

To compute the array of next "k" values of the robot's joint states $[X] = \left[X_{(1)}, X_{(2)}, \dots, X_{(k)}\right]^T$ given an array of "k" control vectors $[U] = \left[U_{(0)}, U_{(1)}, \dots, U_{(k-1)}\right]^T$; it can be done straightforwardly using the equality (9), which uses the initial state vector $X_{(0)}$, the matrix $\mathcal{M}$ and the prediction matrix $\aleph$.

$$[X]_{[m\cdot n\cdot k\times 1]} = \mathcal{M}X_{(0)} + \aleph[U]_{[m\cdot k\times 1]} \tag{9}$$

The matrix $\mathcal{M}$ multiplies the vector of joint's initial state X_0, and it is formed by the stacking of powers of the matrix $\mathbb{A}$ as is shown in the equation (10).

$$\mathcal{M} = \begin{bmatrix} \mathbb{A} \\ \mathbb{A}^2 \\ \vdots \\ \mathbb{A}^k \end{bmatrix}_{[k\cdot n\cdot m\times n\cdot m]} \tag{10}$$

Similarly, the prediction matrix $\aleph$ multiplies the vector $[U]$ composed of the stacking of the control signal vectors for each prediction horizon "k". Also, this matrix is formed by products between powers of the matrix $\mathbb{A}$ with the matrix $\mathbb{B}$ as noted in the equation (11).

$$\aleph = \begin{bmatrix} \mathbb{B} & 0 & \dots & 0 \\ \mathbb{A}\mathbb{B} & \mathbb{B} & \dots & 0 \\ \mathbb{A}^2\mathbb{B} & \mathbb{A}\mathbb{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{A}^{k-1}\mathbb{B} & \mathbb{A}^{k-2}\mathbb{B} & \dots & \mathbb{B} \end{bmatrix}_{[k\cdot n\cdot m \times m\cdot k]} \quad (11)$$

C. Interpolation

If the end-effector of a robotic manipulator has to move from the pose P_0 to the pose P_1 beginning with a velocity V_0 and ending with a velocity V_1 during a period T beginning at t_0, it's important to compute the optimal values of $[U]$ in the equation (9); in such way that the equation (8) valued at $X_{(k)}$, the last element of $[X]$, must meet the conditions of end position and end velocity said.

To being, for the computation of the prediction matrix $\mathbb{M}$, it is important to know Δt and the position derivative order "n". The last one has to do with the desired smoothness of the trajectory. For example, if the derivative of certain order "c" of the joint position is required to be continuous, then the control signal must the following order derivative, this is $n = c + 1$. However, if the derivative of order "c" of the joint position is required to be differentiable in addition to being continuous, then $n = c + 2$. Thereby, for this work to ensure the joint torques to be continuous and differentiable according to the inverse dynamic of Euler-Lagrange equation (11), it can be inferred that joint positions, velocities and accelerations must be continuous and differentiable also, which leads to choose $c = 2$ (second derivative of joint position or acceleration) and then $n = 4$ (fourth derivative of the joint position or jounce).

$$\vec{\tau}_i = M(\vec{q}_i)\ddot{\vec{q}}_i + C(\vec{q}_i, \dot{\vec{q}}_i)\dot{\vec{q}}_i + G(\vec{q}_i) \quad (11)$$

Now, to determine Δt it's necessary to know the number of divisions "k" to be carried out in the period T, so that the fourth derivative of joints remain constant in each interval. To determine the minimum value of "k", it's inferred that $k \geq n$. Consequently, the time interval Δt can be computed by $\Delta t = \frac{T}{k}$, and with this information the matrix $\mathcal{M}$ and the prediction matrix $\aleph$ can be formed using the equations (3), (4), (6), (7), (10) and (11). For this work a value of $k = 6$ was used for a period $T = 2s$.

The loss function (L) to minimize is found depending on the desired posture to be reached P_1, the desired velocity V_1 at that pose, and the vector $Y_{(k)}$, shown in equation (8), which expresses the end-effector final pose and velocity at the end of the path, which is also depending on the time series of the control vectors $[U]$ as posited in equation (9).

$$L(P_1, V_1, [U]) = \left\| \begin{bmatrix} P_1 \\ V_1 \end{bmatrix} - Y_{(k)} \right\|^2 \quad (12)$$

The following step is to compute the set of control actions $[U]$ which can be done by any constrained non-linear minimization strategy as shown in equation (13). For this work, the Quasi-Newton method was used under the constraints highlighted in Table II of joint positions $\vec{q}_i$, velocities $\dot{\vec{q}}_i$ and the torques $\vec{\tau}_i$ computed using the dynamic parameters of the Table III and the equation (11).

$$[U] = \underset{[U]}{\operatorname{argmin}}\{L\} \quad (13)$$

The optimization of (13) is performed under the constrains $\vec{q}_{min} < \vec{q}_{(i)} < \vec{q}_{max}$, $-\vec{v}_l < \dot{\vec{q}}_{(i)} < \vec{v}_l$, and $-\vec{\tau}_l < \vec{\tau}_i < \vec{\tau}_l$. Where $\vec{q}_{min}$ and $\vec{q}_{max}$ are vectors of the lower and maximum values permitted for each of the "n" robot's articulations; $\vec{v}_l$ and $\vec{\tau}_l$ are vectors of the limit values of the joint's velocities and torques permitted.

Finally, once the set of control signals $[U]$ are computed, the interpolation of each joint values x_j is done by integrating n-times each piecewise function $f_j(t)$ formed by each control value $u_{j,i}$ (remembering that $1 \leq j \leq m$ is the joint number, and $1 \leq i \leq k$ is the ith position in time) during the period T as it's shown in equality (14). To do so, the **Cauchy formula for repeated integration** shown in equation (15) was used, considering $1 \leq p \leq n$.

$$f_j(t) = \begin{cases} u_{j,1}: t_0 \leq t < t_0 + \Delta t \\ u_{j,2}: t_0 + \Delta t \leq t < t_0 + 2\Delta t \\ \vdots \\ u_{j,i}: t_0 + (i-1)\Delta t \leq t < t_0 + i\Delta t \\ \vdots \\ u_{j,k}: t_0 + (k-1)\Delta t \leq t < t_0 + T \end{cases} \quad (14)$$

$$\frac{d^{(n-p)}x_j(t)}{dt^{(n-p)}} = \frac{1}{p!}\int_{t_0}^{t}(t-w)^p f_j(w)dw \quad (15)$$

D. Observations

Granted that the loss function (12) has to do with the FK of the robot, this expression if formed by the trigonometric functions' sine and cosine which endow the loss function with non-linearity. For this reason, the optimization is a non-convex problem, although the constrains are linear. Also, because the solution also solves the problem of IK, a problem which has infinite solutions, this is why the solution obtained can be non-global. However, this happens meanly if the system is redundant. Then, if the kinematic chain is non-redundant, the problem can be simplified by using IK to find the final joint values, and then using the Jacobian matrix to calculate the joint velocities. And with these values the only thing that would be left would be to optimize equation (9) where no trigonometric functions are involved, and the problem would be convex. However, the objective of the proposed method is not to use previous solutions of IK, which could bias the results obtained. But IK can be used to check if the target posture is achievable, to rule out problems that have no solution.

On the other hand, the chosen period T must be realistic for the robot to achieve its objective. Opting for an excessively short time might lead to accelerations and torques that violate restrictions, preventing the solver from converging to a solution. It's essential to highlight that this is an interpolation issue, not a control problem. Unlike nonlinear control problems where convergence is assured without specifying the time, in this scenario, the desired time for the robot to reach its goal is predetermined.

III. RESULTS

Considering the manipulator begins its movement with all its joints in the zero position as well as its temporal derivatives. The goal is the end-effector of the manipulator after 2 seconds reaching the task-space pose and the task-space velocity shown in Table IV.

TABLE IV. TARGETS OF THE END-EFFECTOR

Target Pose					
Position (m)			***Orientation (rad)***		
x	y	z	***Pitch*** θ	***Roll*** ζ	***Yaw*** φ
-0.0813	0.2866	0.1322	-0.3131	0.0382	-1.9557
Target Velocity					
Linear (m/s)			***Angular (rad/s)***		
v_x	v_y	v_z	ω_x	ω_y	ω_z
-0.1532	0.1648	0.1582	0.1769	0.0309	0.0487

To give a better illustration of what the previously proposed task represents for the robot, Fig. 2 shows an illustration where ABB-IRB-120 is observed with all its joints at zero which leads to the axes of the end-effector to be perfectly aligned with the axes of the inertial reference frame (red for the x axis, green for the y axis and green for the z axis). The figure also shows the axes of the new target posture of the robot, as well as the vector of the target linear velocity (light blue arrow) and the target angular velocity (purple arrow with oriented arc).

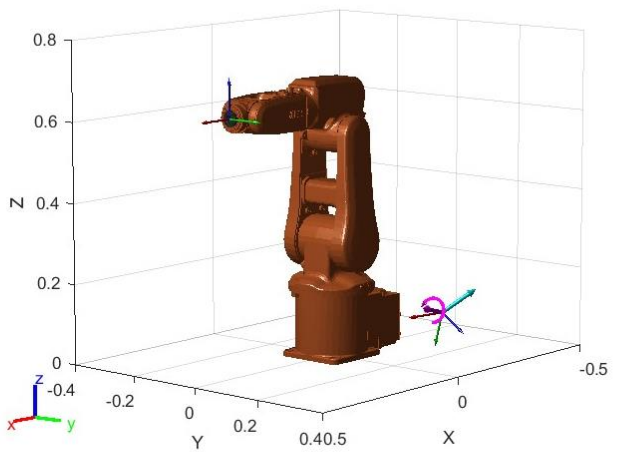

Figure 2. ABB-IRB-120 with its target position and velocity in task-space

Thus, by minimizing the loss function (12) by the constrained Quasi-Newton method, the optimized control signal u_j for each joint along the given period are found. Graphically, it can be appreciated in Fig.3 that after 138 iterations, lasting 5.42s, the cost function reached the minimum value of $6.26611 \cdot 10^{-12}$ which is relatively near zero; meaning that at the end of the given period, the manipulator's end-effector reached the target pose and the target velocity in the task-space, and the optimized control signals allowed this to be achieved.

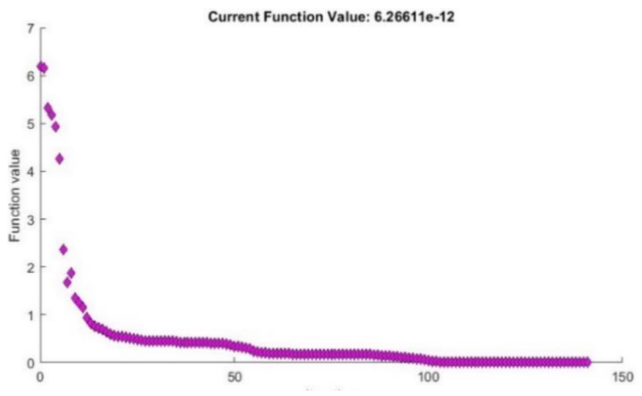

Figure 3. Values of the cost function in each iteration

Then, integrating consecutively these control signals using the formula (15) the interpolation of the joints and their derivatives are arrived. To illustrate this fact, in Fig.4 these results are shown for the first and fourth joints of the robot, where it was graphed the control signals or the fourth derivative of the joint positions (u_1 and u_4) as well as their repetitive integrations over time until reaching the joint values, and also the torques (T_1 and T_4) computed through equation (11). In addition, it can be appreciated that for joint positions, velocities and torques, their values are kept within the boundaries (red lines) given in Table II.

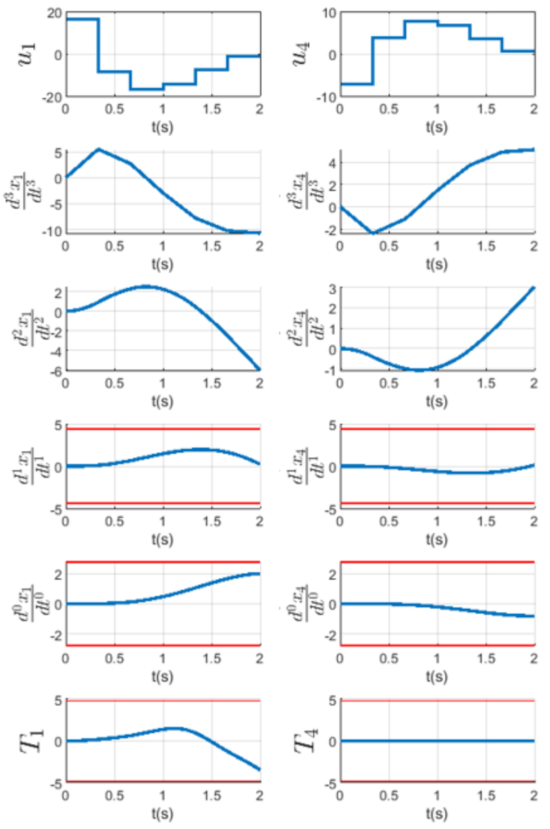

Figure 4. Control signals, derivatives and torques of the first and fourth manipulator's joint and its torques

Finally, with the joint values over time computed, these values can be used in the FK equation of the robot to recreate the trajectory in the task-space that the robot's end effector must travel during the given period. The result of this operation is shown in Fig.5, where it can be appreciated the path and the axes of the end-effector, which corroborates that this reached the objective of its trajectory at the end.

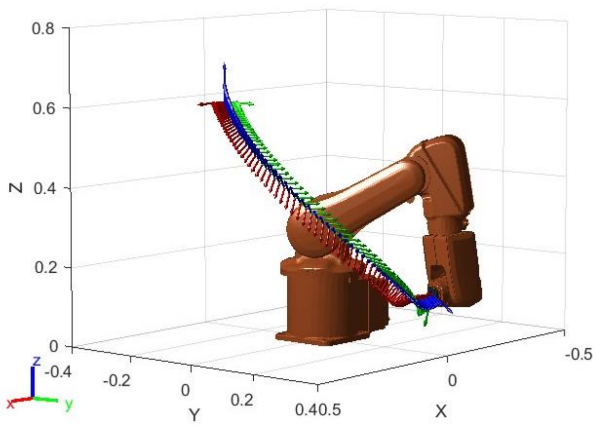

Figure 5. Result of the interpolation in the task-space of the robot ABB-IR-120

IV. Conclusions

In summary, the methodology outlined in this study successfully executed trajectory planning within the joint-space of the ABB-IRB-120 robotic model without prior knowledge of the inverse kinematics solution for the given targets in the task-space. This was achieved by modeling the kinematics of the robot considering the joint positions and their derivatives for the state vector, and the direct together with the differential kinematics for the output vector of the system. Thus, by defining the specified target posture and target velocity of the end-effector in the task-space; the approach seamlessly computed the control signals of the joints within the predefined motion duration using the constrained Quasi-Newton method, just having a low error of $6.26611 \cdot 10^{-12}$ on the manipulator's end-effector pose and velocity with respect to the previous defined task-space target at the end of the path. Crucially, the method took into account both kinematic constraints, encompassing joint positions and velocities considerations, and dynamic constraints related to joint torques. The obtained results not only guaranteed the continuity and differentiability up to the second derivative (C_2 curves) in both task-space and joint-space trajectories but also ensured continuity up to the third derivative.

References

[1] Akira, Terui., Masahiko, Mikawa. (2023). Inverse kinematics and path planning of manipulator using real quantifier elimination based on Comprehensive Gröbner Systems. arXiv.org, abs/2305.12451 doi: 10.48550/arXiv.2305.12451

[2] Jingdong, Zhao., Xiao, Yang., Zhiyuan, Zhao., Guocai, Yang., Liang, Zhao. (2023). Inverse Kinematics and multi-objective configuration optimization of the SSRMS manipulator. Advances in Space Research, doi: 10.1016/j.asr.2023.06.058

[3] Mrunal, Kanti, Mishra., Sambaran, Ghosal., Arun, Kumar, Samantaray., Goutam, Chakraborty. (2020). Jacobian-Based Inverse Kinematics Analysis of a Pneumatic Actuated Continuum Manipulator. doi: 10.1007/978-981-16-1769-0_1

[4] Alkayyali, M., & Tutunji, T. A. (2019). PSO-based algorithm for inverse kinematics solution of robotic arm manipulators. https://doi.org/10.1109/rem.2019.8744103

[5] Dereli, S., & Köker, R. (2019). A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm. Artificial Intelligence Review, 53(2), 949–964. https://doi.org/10.1007/s10462-019-09683-x

[6] Momani, S., Abo-Hammour, Z. S., & Alsmadi, O. M. (2016). Solution of Inverse Kinematics Problem using Genetic Algorithms. Applied Mathematics & Information Sciences, 10(1), 225–233. https://doi.org/10.18576/amis/100122

[7] Lee, C., & Chang, J. (2021). A Workspace-Analysis-Based genetic algorithm for solving inverse kinematics of a Multi-Fingered anthropomorphic hand. Applied Sciences, 11(6), 2668. https://doi.org/10.3390/app11062668

[8] Aristidou, A., & Lasenby, J. (2011). FABRIK: A fast, iterative solver for the Inverse Kinematics problem. Graphical Models, 73(5), 243–260. https://doi.org/10.1016/j.gmod.2011.05.003

[9] Moreno, L. a. O., & Alcantara, J. H. A. (2022). An adaptation of FABRIK algorithm for serial robot's inverse kinematics. https://doi.org/10.1109/icev56253.2022.9959663

[10] Santos, M. F., Molina, L., Carvalho, E. a. N., Freire, E. O., Carvalho, J. C. A., & Santos, P. C. (2021). FABRIK-R: an extension developed based on FABRIK for robotics manipulators. IEEE Access, 9, 53423–53435. https://doi.org/10.1109/access.2021.3070693

[11] Dong, G., Huang, P., Wang, Y., & Li, R. (2022). A modified forward and backward reaching inverse kinematics based incremental control for space manipulators. Chinese Journal of Aeronautics, 35(12), 287–295. https://doi.org/10.1016/j.cja.2021.08.014

[12] Zou, Le., Zhize, Wu., Chen, Zhang., Xiao-Feng, Wang., Ding, Zesheng., Tan, Ming. (2020). Path planning method and system based on parameterized Thiele continued fraction interpolation.

[13] Yihua, Hu., Shulin, Zhang., Yanhui, Chen. (2023). Trajectory planning method of 6-DOF modular manipulator based on polynomial interpolation. Journal of Computational Methods in Sciences and Engineering, doi: 10.3233/jcm-226672

[14] Zhe, Liu., Aiqiang, Pan., Anfeng, Jiang., Wenhe, Li., Jiawei, Zhang., Chengchao, Bai. (2022). Research on Motion Path Planning Method of Live Working Robot Manipulator Based on Reverse Splitting Calculation. Journal of physics, doi: 10.1088/1742-6596/2213/1/012031

[15] Tae-Won, Kang., Jin-Gu, Kang., Jin-Woo, Jung. (2021). A Bidirectional Interpolation Method for Post-Processing in Sampling-Based Robot Path Planning. Sensors, doi: 10.3390/S21217425

[16] Kenwright, B. (2023). Dual-Quaternion interpolation. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2303.13395

[17] Pu, Y., Shi, Y., Lin, X., Hu, Y., & Li, Z. (2020). C2-Continuous orientation planning for robot End-Effector with B-Spline curve based on logarithmic quaternion. Mathematical Problems in Engineering, 2020, 1-16. https://doi.org/10.1155/2020/2543824

[18] Giovanni, Legnani., Giovanni, Legnani., Irene, Fassi., Alessandro, Tasora., Dario, Fusai. (2021). A practical algorithm for smooth interpolation between different angular positions. Mechanism and Machine Theory, doi: 10.1016/J.MECHMACHTHEORY.2021.104341

[19] Ahmad, Taher, Azar., Fernando, E., Serrano., Nashwa, Ahmad, Kamal., Anis, Koubaa., Adel, Ammar., Ibraheem, Kasim, Ibraheem., Amjad, J., Humaidi. (2021). Finite Element Euler-Lagrange Dynamic Modeling and Passivity Based Control of Flexible Link Robot.. doi: 10.1007/978-3-030-76346-6_41

[20] Balgaisha, Mukanova. (2020). Control of Actuators Torques for Optimal Movement along a Given Trajectory for the DexTAR Robot. Applied and Computational Mechanics, doi: 10.22055/JACM.2020.34650.2449

[21] Barhaghtalab, M. H., Meigoli, V., Haghighi, M. R. G., Nayeri, S. A., & Ebrahimi, A. (2018). Dynamic analysis, simulation, and control of a 6-DOF IRB-120 robot manipulator using sliding mode control and boundary layer method. Journal of Central South University, 25(9), 2219-2244. https://doi.org/10.1007/s11771-018-3909-2

[22] Gregory, A, DiLisi. (2019). Classical Mechanics, Volume 2.