A Comparison between Linear, Nonlinear, and Adaptive MPC Controllers for ABB IRB120 Robot in Collaborative Assembly

Hend M. Aafia
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
hend.aafia@gmail.com

Kerlos R. Zakhary
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
kerols.rady@gmail.com

Reem H. Gheith
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
17p8047@eng.asu.edu.eg,

Ahmed M. Gomaa
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
1500194@eng.asu.edu.eg

Antonios A. Saleh
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
antonios_abouelkher@outlook.com

Verina G.Guirgis
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
17p8106@eng.asu.edu.eg

Mohammed I. Awaad
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
Mohammed.awad@eng.asu.edu.eg

Shady A. Maged.
Mechatronics Engineering Department,
Ain Shams University.
Cairo, Egypt
Shady.maged@eng.asu.edu.eg

2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC) | 978-1-6654-6677-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/MIUCC55081.2022.9781651

***Abstract*— Collaborative Assembly is becoming more important each day because of the vast advances that the world is witnessing due to Industry 4.0. This paper presents a model-based control architecture for the ABB IRB120 robot. This robot is 6-DOF which imposes huge nonlinearity in system modeling. An extensive study on the robot kinematics followed by the design of Linear MPC, Nonlinear MPC, and Adaptive MPC are introduced as possible control architectures for the position of the robot's end-effector. The controllers in this paper are designed to allow the robot to reach certain set-points with the best possible precision considering the dynamic limits of the robot.**

Keywords—ABB IRB120 Robot, Model Predictive Controller, Linearization, Discretization, Linear MPC, Nonlinear MPC, Adaptive MPC, PID.

I. INTRODUCTION

Nowadays, the collaboration between robots and humans is beginning to be introduced to most industries. And a considerable concern is being paid to the control design concerned with the position of the robot end-effector (1). In collaborative assembly, robots are required to pick and place objects and hand them to the human in the loop. Thus, the robot is needed to reach a certain set-point without paying attention to its path. In this case, the controller's efficiency is measured by the system's time response (Fig. 5-25). In our system, the multipurpose industrial robot of 6-DOF, ABB IRB-120, is used. Studies have been made to develop the robot model. However, they face the challenge of its complex nonlinear behavior. This paper introduces several methods, including linear, nonlinear, and adaptive model predictive controllers. However, these controllers face the challenge of compromising between the high computational resources needed and the reliability and efficiency of the controller. Motivated by these two challenges, this paper discusses the three controllers, linear MPC with its simple linearized models; nonlinear MPC and the required formulation of its model; and finally, adaptive MPC and its model updating strategy with excessive linearization and discretization. This paper goes through the nonlinear model of the ABB IRB10 robot. Then model linearization and discretization are discussed. Finally, the paper uses three MPC types to control the end-effector position (1).

II. STATE SPACE MODEL CONSTRUCTION AND DISCRETIZATION:

This section presents the model for ABB IRB 120 robot. A kinematic model is suggested since it is assumed that the robot is not required to move at high velocities in most collaborative assembly tasks. The end effector's position in the base frame is represented as:

$$p_e = \begin{pmatrix} p_x \\ p_y \\ p_z \\ \emptyset \\ \theta \\ \Psi \end{pmatrix} \quad (1)$$

ABB IRB120 robot has 6 DOF and is referred to as the joints space and represented by (2) where q_i represents the rotation of the joint i (Fig. 1):

$$q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{pmatrix} \quad (2)$$

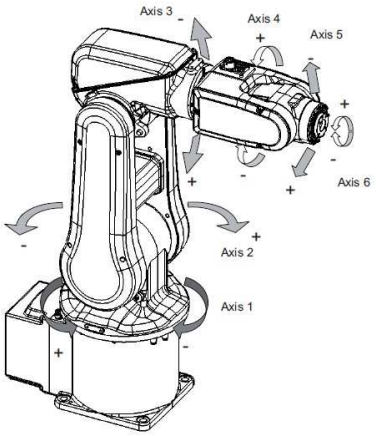

Fig. 1: ABB IRB120 DOF

978-1-6654-6677-6/22/$31.00 ©2022 IEEE

A kinematic model is used to map joint space into cartesian space. It is obtained using the D-H (Denavit-Hartenberg) parameters. Then, Jacobian is used as the relationship between end-effector velocity and the velocity of the joints as:

$$v_e = J(q) \times \dot{q} \tag{3}$$

Where the end-effector velocity (v_e) is a vector of a size 6×1 and joint velocities ($\dot{q}$) is also of a size 6×1. Thus, the Jacobian matrix is of a size 6×6.

Accordingly, a state-space model is constructed where:

The states of the system are the velocity of the end-effector as:

$$x = \begin{pmatrix} x_e \\ y_e \\ z_e \\ \alpha_e \\ \beta_e \\ \gamma_e \end{pmatrix} \tag{4}$$

The input of the system is the velocities of the joints as:

$$U = \begin{pmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \\ \dot{q_5} \\ \dot{q_6} \end{pmatrix} \tag{5}$$

And the output of the system is:

$$Y = \begin{pmatrix} x_e \\ y_e \\ z_e \\ \alpha_e \\ \beta_e \\ \gamma_e \end{pmatrix} \tag{6}$$

A continuous model is developed as:

$$\begin{aligned} A_{cont} &= (0)_{6\times6} \\ B_{cont} &= J_{6\times6} \\ C_{cont} &= I_{6\times6} \\ D_{cont} &= (0)_{6\times6} \end{aligned} \tag{7}$$

This model is discretized for a sampling time of 10^{-2} as:

$$e^{\begin{pmatrix} A_{cont6\times6} & B_{cont6\times6} \\ (0)_{6\times6} & (0)_{6\times6} \end{pmatrix}\times T} = \begin{pmatrix} A_d & B_d \\ 0 & I \end{pmatrix} \tag{8}$$

$$\begin{aligned} A_{dis} &= (I_{6\times6}) \\ B_{dis} &= J_{6\times6} \times T \\ C_{dis} &= I_{6\times6} \\ D_{dis} &= (0)_{6\times6} \end{aligned} \tag{9}$$

III. CONTROL DESIGN

A. MPC Cost function:

In this section, our control schemes are discussed. The implementation and results of three types of Model Predictive Controllers are compared. Model Predictive Controllers calculate the control actions for each prediction horizon using model-based predictions. MPCs utilize a cost function to find the optimal sequence of control actions for a particular prediction horizon; the cost function used is QP (Quadratic Program) functions as:

$$J(z_k) = J_y(z_k) + J_u(k_z) + J_{\Delta u}(z_k) + J_\varepsilon(z_k) \tag{10}$$

Where z_k is the QP decision, and it is a sequence of manipulated variable values for the upcoming control intervals for the next prediction horizon and given by:

$$z_k^T = [u(k|k)^T \quad u(k+1|k)^T \cdots \quad u(k+p-1|k)^T \quad \varepsilon_k]. \tag{11}$$

Where k is the current control interval, p is the prediction horizon (number of intervals) and ε_k is the slack variable used for constraint softening.

The first term in the cost function is $J_y(z_k)$ representing Output Reference Tracking, and it tracks how far the plant output is from a certain set-point. And given by:

$$J_y(Z_k) = \sum_{j=1}^{n_y} \sum_{i=1}^{P} \{w_{yj} [\, rj\,(k+i\,|k) - yj\,(k+i\,|k)]\}^2 \tag{12}$$

Where n_y is the number of plant outputs, w_{yj} is the weight of the j^{th} output, $r_j(k+i\,|k)$ represents the set-point of the j^{th} output during the i^{th} interval, $y_j(k+i\,|k)$ is the predicted j^{th} plant during the i^{th} interval.

The second term is J_u representing Manipulated Variable Tracking, and it tracks how far the manipulated variable is from its nominal value. In our case, this nominal value is zero to minimize the consumed power. J_u can be obtained by:

$$J_u = \sum_{j=1}^{n_u} \sum_{i=0}^{P-1} \{\, w_{uj} [\, uj\,(k+i\,|k) - uj_{nomi}\,(k+i\,|k)\,]\}^2 \tag{13}$$

Where n_u is the number of plant inputs, w_{uj} is the weight of the j^{th} input, $u_j(k+i\,|k)$ represents the value of the j^{th} input during the i^{th} interval, $u_{j_{nom}}(k+i\,|k)$ is the nominal value of j^{th} input during the i^{th} interval.

The third term is $J_{\Delta u}$ representing Manipulated Variable Move Suppression. It is essential to ensure small changes in the manipulated variables, and it is more critical to ensure safety in collaborative tasks. This term can be obtained as:

$$J_{\Delta u} = \sum_{j=1}^{n_u} \sum_{i=0}^{P-1} \{w_{\Delta uj} [\, uj(k+i\,|k) - uj(k+i-1\,|k)]\}^2 \tag{14}$$

Where n_u is the number of plant inputs, $w_{\Delta uj}$ is the weight of the j^{th} input rate, uj(k+i |k)-uj(k+i-1 |k) represents the change in the j^{th} input from the previous interval.

Finally, J_ε is the Constraint Violation, a soft representation of the system's constraints. And can be calculated as:

$$J_\varepsilon = \rho_\varepsilon \times \varepsilon_k^2 \quad (15)$$

Where ρ_ε is the constraint violation penalty weight.

B. Linear MPC:

Linear MPC is based on linearizing and discretizing the plant model only once near the equilibrium points. The control architecture is shown in (Fig. 2):

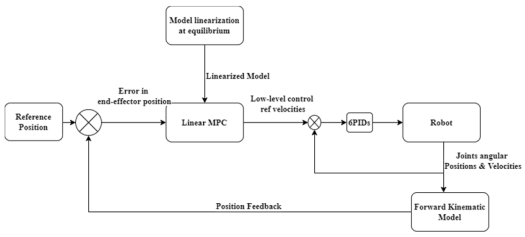

Fig. 2: Linear MPC Block Diagram

Based on the model discussed in section (II), a MIMO state-space model is constructed after linearization using Taylor series expansion. In this case of linear MPC, the plant linearization is performed only once for each equilibrium point.

C. Nonlinear MPC:

Nonlinear MPC is implemented for the actual nonlinear state-space model of the plant. It solves the MPC optimization problem for this nonlinear model. It utilizes two functions to represent the state and output equations of the state-space model of the plant. Accordingly, huge computational resources are required to solve the MPC optimization problem. The block diagram of this controller is shown in (Fig. 3):

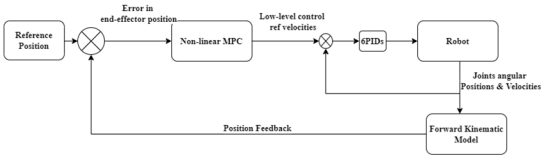

Fig. 3: Nonlinear MPC Block Diagram

D. Adaptive MPC:

Adaptive MPC is based on linearizing and discretizing the plant model, then updating this model each sample time to solve the optimization problem. It benefits from the simplicity of linear MPC's optimization function and the general model of nonlinear MPC. The control architecture is shown in (Fig. 4):

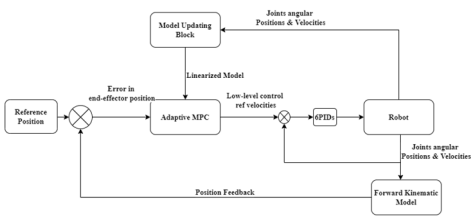

Fig.4: AMPC Block Diagram

The key element of this control strategy is the model updating function. This function takes information about joints' velocities and positions and updates the linearized model each sample time.

E. PID Controllers:

Our low-level control is implementer using six regular PIDs. These controllers are SISO controllers. And they are mainly used to actuate the motors of the robot joints to reach the output reference velocities produced by the high-level adaptive MPC.

IV. Results and Discussion

In this section, we simulated the performance of the three controllers. MATLAB Simulink libraries were utilized to solve the MPC optimization problem. A comparison using rise time, overshooting, accuracy, and RMSE is carried out. In addition, a measure of the actual real-time simulation is done as an indicator of the computational power needed. MATLAB MPC Tuner was used at the beginning to select the controller parameters, then fine trial and error tunning was used. The model parameters are shown in (Tables 1-2).

TABLE 1: MPC PARAMETERS

Parameter	Value
Sampling time	0.01
Prediction Horizon	5
Control Horizon	3
MV Nominal Value	[0 , 0 , 0 , 0 , 0 , 0]
MV Weights	[1.5 , 1.5 , 1.5 , 1.5 , 1.5 , 1.5]
Manipulated Variable Rate Weights	[0.044 , 0.044 , 0.044 , 0.044 , 0.044 , 0.044]
Output Variable Weights	[90 , 90 , 90 , 50 , 50 , 50]
ECR	500

TABLE 2: PLANT CONSTRAINS

Parameter	Max	Min
MV1	250 $(°/s)$	-250 $(°/s)$
MV2	250 $(°/s)$	-250 $(°/s)$
MV3	250 $(°/s)$	-250 $(°/s)$
MV4	320 $(°/s)$	-320 $(°/s)$
MV5	320 $(°/s)$	-320 $(°/s)$
MV6	420 $(°/s)$	-420 $(°/s)$
MV1 Rate	10 $(°/s^2)$	-10 $(°/s^2)$
MV2 Rate	10 $(°/s^2)$	-10 $(°/s^2)$
MV3 Rate	10 $(°/s^2)$	-10 $(°/s^2)$
MV4 Rate	10 $(°/s^2)$	-10 $(°/s^2)$
MV5 Rate	10 $(°/s^2)$	-10 $(°/s^2)$
MV6 Rate	10 $(°/s^2)$	-10 $(°/s^2)$
1st Output Variable	0.58	-0.56
2nd Output Variable	0.58	-0.58
3rd Output Variable	0.85	-0.112
4th Output Variable	180 (°)	-180 (°)
5th Output Variable	180 (°)	-180 (°)
6th Output Variable	180 (°)	-180 (°)

The results are based on the same set-point. The response of the system was observed and compared.

A. Linear MPC:

The real-time needed by the solver to solve this optimization problem is 5.13 seconds

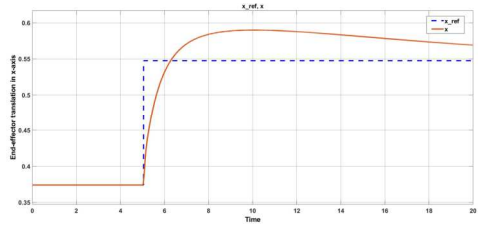

Fig. 5: End-effector translation in x-axis (X_e)

The robot's translation on the x-axis is characterized by an overshoot of 7.72%, a rise time of 1.762 sec, an accuracy of 96.06% within 15 sec, and RMSE of 0.033 m.

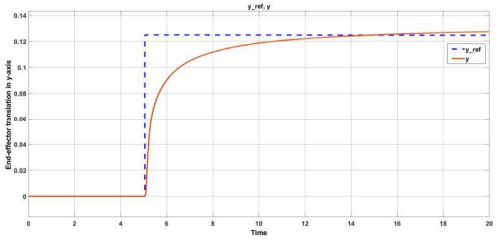

Fig. 6: End-effector translation in the y-axis (Y_e)

The robot's translation on the y-axis is characterized by an overshoot of 0.508%, a rise time of 3.558 sec, an accuracy of 98.02% within 15 sec, and RMSE of 0.0162.

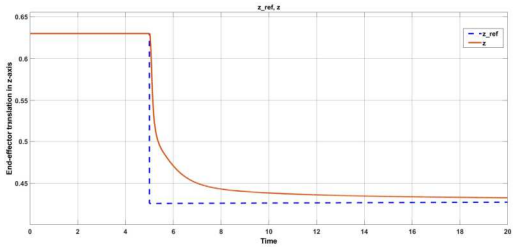

Fig. 7: End-effector translation in z-axis(Z_e)

The robot's translation on the z-axis is characterized by an overshoot of 0.06%, a rise time of 2.308 sec, an accuracy of 98.41% within 15 sec, and RMSE of 0.0233.

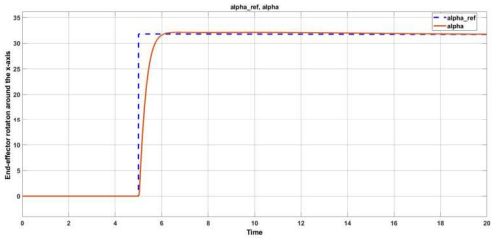

Fig. 8: End-effector rotation around the x-axis (Alpha_e)

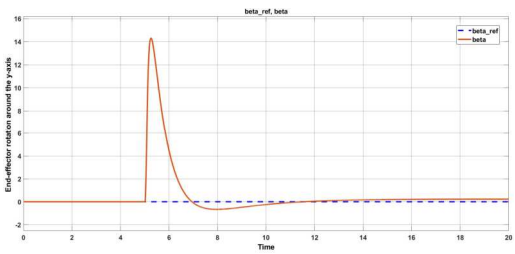

Fig. 9: End-effector rotation around the y-axis (Beta_e)

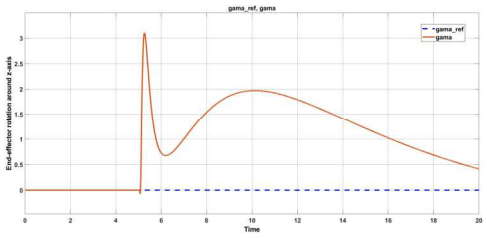

Fig. 10: End-effector rotation around the z-axis (Gamma_e)

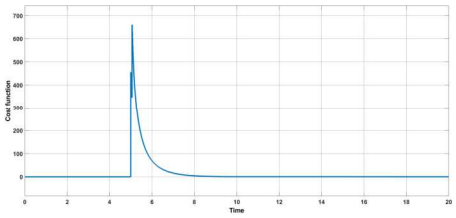

Fig. 11: Cost Function

B. Nonlinear MPC:

The real-time needed to solve this optimization problem is 16 mins and 15 sec.

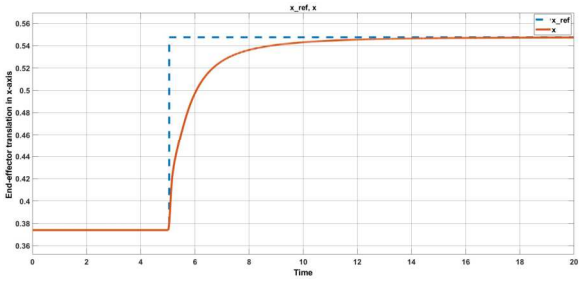

Fig. 12: End-effector translation in x-axis (X_e)

The robot's translation on the x-axis is characterized by an overshooting of 0.336%, a rise time of 2.145 sec, an accuracy of 99.97% within 15 sec, and RMSE of 0.0222m.

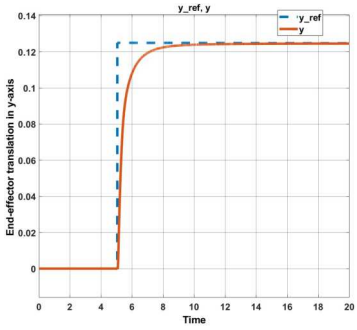

Fig. 13: End-effector translation in the y-axis (Y_e)

The robot's translation on the y-axis is characterized by an overshooting of 0.134%, a rise time of 1.053 sec, an accuracy of 99.85% within 15 sec, and RMSE of 0.0130.

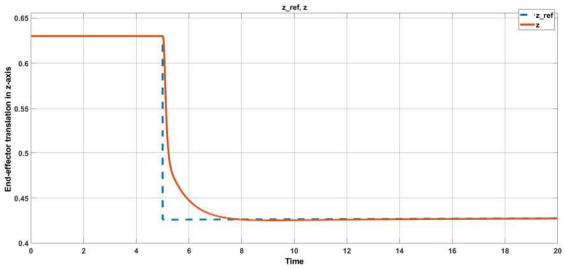

Fig. 14: End-effector translation in z-axis(Z_e)

The robot's translation on the z-axis is characterized by an overshooting of 0.508%, a rise time of 1.273 sec, an accuracy of 99.97% within 15 sec, and RMSE of 0.0175m.

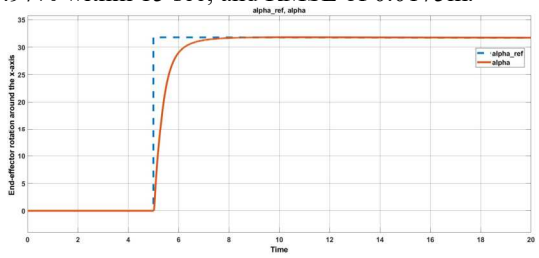

Fig. 15: End-effector rotation around the x-axis (Alpha_e)

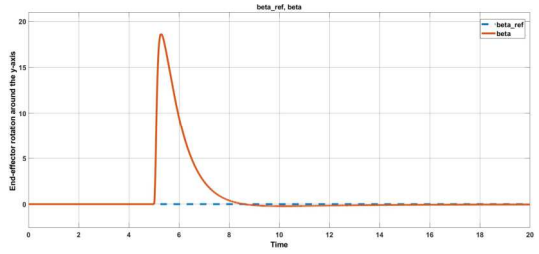

Fig. 16: End-effector rotation around the y-axis (Beta_e)

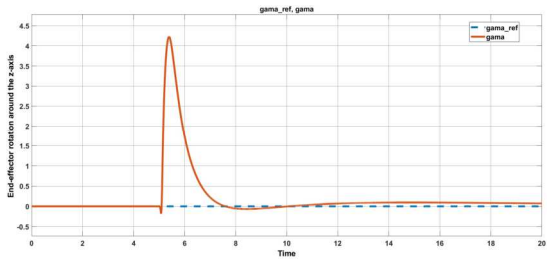

Fig. 17: End-effector rotation around the z-axis (Gamma_e)

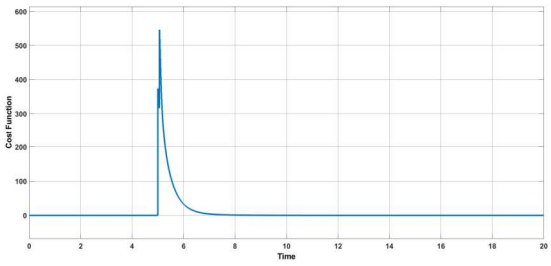

Fig. 18: Cost Function

C. *Adaptive MPC:*

The real-time needed to solve the optimization problem for Adaptive MPC is 9.48 seconds

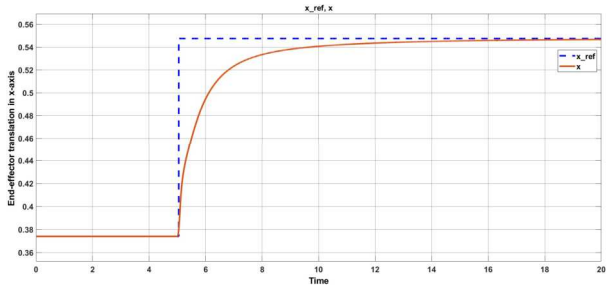

Fig. 19: End-effector translation in x-axis (X_e)

The robot's translation in the x-axis is characterized by an overshooting of 0.312%, a rise time of 2.343 sec, an accuracy of 99.85% within 15 sec, and RMSE of 0.0230 m.

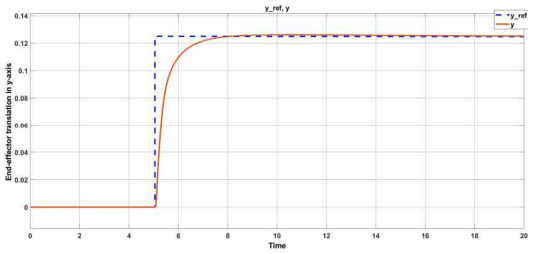

Fig. 20: End-effector translation in the y-axis (Y_e)

The robot's translation in the y-axis is characterized by an overshooting of 0.505%, a rise time of 1.039 sec, an accuracy of 99.82% within 15 sec, and RMSE of 0.012.

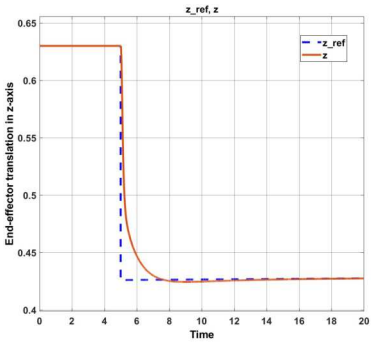

Fig. 21: End-effector translation in z-axis(Z_e)

The robot's translation in the z-axis is characterized by an overshooting of 1.998%, a rise time of 0.977 sec, an accuracy of 99.67% within 15 sec, and RMSE of 0.0179

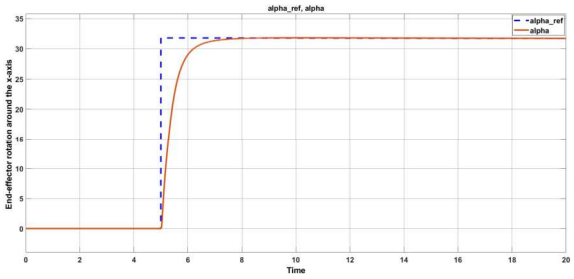

Fig. 22: End-effector rotation around the x-axis (Alpha_e)

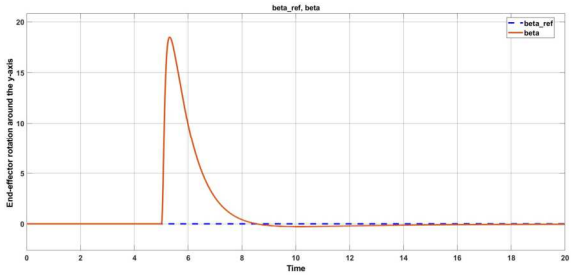

Fig. 23: End-effector rotation around the y-axis (Beta_e)

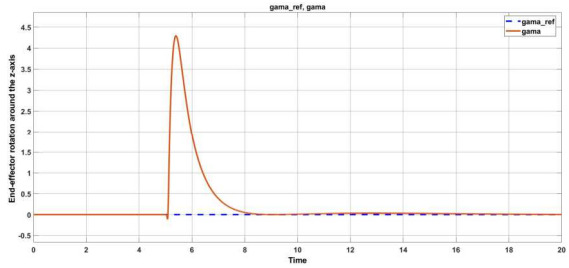

Fig. 24: End-effector rotation around the z-axis (Gamma_e)

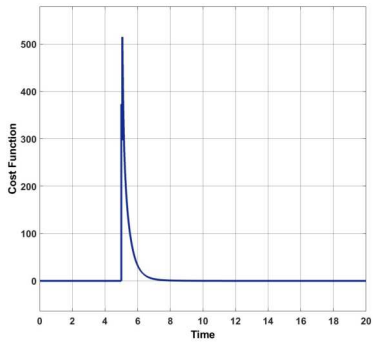

Fig. 25: Cost Function

TABLE 3: CONTROLLERS RESULTS

Parameter	Linear MPC	Nonlinear MPC	Adaptive MPC
Solution total Time	5.13 sec.	975 sec.	9.48 sec.
Overshoot in x-axis	7.72%	0.336%	0.312%
Rise time in x-axis	1.762 sec.	2.145	2.343 sec
Accuracy in x-axis	96.06%	99.97%	99.85%
RMSE in x-axis	0.003 m	0.0222m	0.023 m
Overshoot in y-axis	0.508%	0.134%	0.505%
Rise time in y-axis	3.558 sec.	1.053 sec	1.039 sec
Accuracy in y-axis	98.02%	99.85%	99.82%
RMSE in y-axis	0.0162 m	0.013 m	0.012 m
Overshoot in z-axis	0.06%	0.508%	1.998%
Rise time in z-axis	2.308 sec.	1.273 sec	0.977 sec
Accuracy in z-axis	98.41%	99.97%	99.67%
RMSE in z-axis	0.0233 m	0.0175 m.	0.0179 m

V. CONCLUSION

This paper introduces a brief about the ABB IRB-120 robot model. A nonlinear state-space model was presented, then linearized and discretized. Finally, a comparison between three types of model predictive controllers is discussed. The results show that Linear MPC is the fastest and simplest controller; however, it is the lowest accuracy. This is because of the simple linear model used to solve the optimization problem. Nonlinear MPC is the highest in accuracy and lowest in RMSE; however, it requires huge computational resources. As nonlinear MPC solves the optimization problem for a nonlinear plant, accordingly, better predictions are obtained. Finally, adaptive MPC balances the high accuracy and the low computational power needed as it solves an optimization function for a linear model that's updated many times during the solution. Adaptive MPC is computationally more efficient than nonlinear MPCs yet more reliable than linear MPCs for such an extremely nonlinear system.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] DE GIER, M. R. Control of a robotic arm: Application to on-surface 3D-printing. 2015.

[3] HADI BARHAGHTALAB, Mojtaba, et al. 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, 2018, 25.9: 2219-2244.

[4] CRISTOIU, Cozmin; NICOLESCU, Adrian. New approach for forward kinematic modeling of industrial robots. Res. & Sci. Today, 2017, 13: 136.

[5] TUTSOY, Onder, et al. Developing linear and nonlinear models of ABB IRB120 industrial robot with Maplesim multibody Modelling software. The Eurasia Proceedings of Science Technology Engineering and Mathematics, 2017, 1: 273-285.

[6] ROBOTICS, A. B. B. Product specification IRB 120. Document ID: 3HAC05960-001 Revision: Q, 2021.

[7] G. Xia, Z. Xiao and P. Ji, "ABB-IRB120 Robot Modeling and Simulation Based on MATLAB," 2022 International Seminar on Computer Science and Engineering Technology (SCSET), 2022, pp. 23-26, doi: 10.1109/SCSET55041.2022.00015.

[8] PEREIDA, Karime; SCHOELLIG, Angela P. Adaptive model predictive control for high-accuracy trajectory tracking in changing conditions. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018. p. 7831-7837.