Exploring Nonlinear Model Predictive Control in Two-Link Robot Arm Motion Planning

1st Amr Marey
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Canada
amarey@ualberta.ca

***Abstract*—This paper explores the application of nonlinear model predictive control (MPC) for motion planning in a two-link rotating robot arm. The study focuses on improving the efficiency and effectiveness of motion trajectories in robotic arms utilizing an MPC strategy. The research first derives the dynamics of the robot arm. It then applies discrete-time MPC to optimize trajectory outcomes based on estimations of the model dynamics and external input. We explore two different methods to MPC. The first method revolves around the use of a long prediction horizon to approximate the infinite-horizon MPC scenario. The second method uses terminal-state constraints to modify the control in shorter prediction horizon scenarios. The outcomes of the experiments show that both strategies are efficient and that they have distinct benefits in terms of control accuracy and computing demand. The report also describes future improvements that could be made, such as integration with more sophisticated robot models and real-time application. This work lays the groundwork for future investigations for nonlinear MPC-based motion planning.**

***Index Terms*—motion planning, optimal control, model predictive control, optimization**

I. Introduction

Motion planning (MP) for robotic manipulators is a very popular field of study over the past few decades. It is typically characterized by the process of designing robot trajectories and control laws so they can navigate their environments and go from one initial state to a goal state while dodging obstacles and optimizing a certain aspect associated with the path [1]. Some of these optimization objectives include: minimizing trajectory-time, minimizing energy consumption, and more. One can find a survey with regards to MP in [2].

There are multiple MP algorithms presented in the literature. Sampling-based MP algorithms such as Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) are well established within the literature [3]. PRM builds an environment road-map by randomly sampling points in the configuration space, checking for collisions, and directly connecting surrounding nodes if possible. RRT extends a tree rooted at the start configuration using random samples from the configuration space. It tends to work well in single-query scenarios. RRT's ease of use and capacity for swiftly navigating vast areas make it a popular choice in the field [3].

Genetic Algorithms (GA) have also been employed for the MP problem [4]. GAs are a class of search and optimization methods based on the ideas of natural selection and genetics. These algorithms start with a population of feasible path solutions and gradually refine these paths, based on their fitness functions, and choosing paths based on how well they satisfy predetermined optimization criteria. They can also be integrated with other MP algorithms such as sampling-based MP algorithms [4].

Another class of MP algorithms are graph search (GS) algorithms [1]. The basic concept underlying GS methods in motion planning is to visualize the MP problem as a graph, with nodes denoting potential environmental states or configurations and edges denoting possible transitions or motions between these states [2]. The robot's surroundings and potential moves are modelled by a graph that the algorithm creates. Every node in the graph represents a particular configuration or state, and every edge denotes a practicable motion or action that can change the robot's configuration. The algorithm then explores the graph using a search method, such as depth-first, breadth-first, or A* search until finds a suitable path [1].

Another class of MP algorithms involve the use of optimal control (OP). OP techniques concentrate on producing the most effective paths to move from an initial state while respecting different state constraints and minimizing a cost function [5]. Model predictive control (MPC) builds on OP by forecasting the future states of the system over a finite horizon using a model of the system dynamics and external inputs. A cost function is optimized at each single stage to determine the optimal input trajectory. Only the first few values of this input trajectory are applied to the real robot and the optimization process is repeated again. This allows the MP algorithm to be more robust and adaptive to different to scenarios [6].

This paper aims to explore the usage of nonlinear MPC on the two-link rotating robot-arm. The rest of this paper is divided as follows: Section II derives the continuous and discrete-time dynamics of the robot-arm. Section III discusses the basics of MPC and two different MPC approaches one can use for the MP problem. Section IV presents the results and discussions of these two different MPC approaches. Recommendations for future work are presented in V. Section VI concludes this paper.

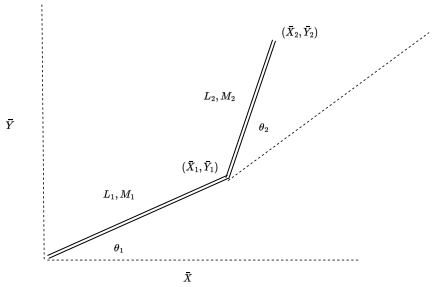


Fig. 1: A simple illustration of the two-link robot arm.

II. Modeling Process

A. Lagrangian Formulation of Two-Link Robot Arm Dynamics

A simple illustration of the two-link robot arm is presented in Fig. 1. θ_i, L_i, M_i and u_i are the joint angles, lengths, masses, and applied input torques of the first $(i = 1)$ and second $(i = 2)$ links respectively. g denotes the gravitational acceleration constant. $(\bar{X}_i, \bar{Y}_i)$ describes the Cartesian coordinates at the end of each joint. This paper will utilize the Lagrangian formulation of the two-link robot-arm. It is assumed that the robot dynamics are ideal (there is no friction or any other non-idealities).

Through simple trigonometry, The kinematic equations for each of the joint ends can be obtained:

$$\begin{pmatrix} \bar{X}_1 \\ \bar{Y}_1 \end{pmatrix} = \begin{pmatrix} L_1 \cos(\theta_1) \\ L_1 \sin(\theta_1) \end{pmatrix}, \tag{1}$$

$$\begin{pmatrix} \bar{X}_2 \\ \bar{Y}_2 \end{pmatrix} = \begin{pmatrix} L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \end{pmatrix}. \tag{2}$$

The velocities of each of the joints is

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} \sqrt{\dot{\bar{X}}_1^2 + \dot{\bar{Y}}_1^2} \\ \sqrt{\dot{\bar{X}}_2^2 + \dot{\bar{Y}}_2^2} \end{pmatrix}. \tag{3}$$

Where

$$\begin{pmatrix} \dot{\bar{X}}_1 \\ \dot{\bar{Y}}_1 \\ \dot{\bar{X}}_2 \\ \dot{\bar{Y}}_2 \end{pmatrix} = \begin{pmatrix} -L_1 \dot{\theta}_1 \sin(\theta_1) \\ L_1 \dot{\theta}_1 \cos(\theta_1) \\ -L_1 \dot{\theta}_1 \sin(\theta_1) - L_2 \dot{\theta}_2 \sin(\theta_2) \\ L_1 \dot{\theta}_1 \cos(\theta_1) + L_2 \dot{\theta}_2 \cos(\theta_2) \end{pmatrix}. \tag{4}$$

The total kinetic energy in all the joints E_{KE} is described by

$$E_{KE} = \frac{1}{2} M_1 V_1^2 + \frac{1}{2} M_2 V_2^2 \tag{5}$$

The total potential energy in all the joints E_{PE} is described by

$$E_{PE} = M_1 g \bar{Y}_1 + M_2 g \bar{Y}_2. \tag{6}$$

The Euler-Lagrange equation obtained from Lagrangian dynamics [7] is

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}\right) - \frac{\partial \mathcal{L}}{\partial \theta_i} = u_i, \quad \mathcal{L} = E_{KE} - E_{PE}. \tag{7}$$

Once can combine (1)-(7) to yield

$$\begin{aligned} &(M_1 + M_2) L_2 \ddot{\theta}_1 + M_2 L_1 L_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &+ M_2 L_1 L_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + (M_1 + M_2) g L_1 \cos(\theta_1) = u_1, \end{aligned} \tag{8}$$

$$\begin{aligned} &M_2 L_2^2 \ddot{\theta}_2 + M_2 L_1 L_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &\quad - M_2 L_1 L_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + M_2 g L_2 \cos(\theta_2) = u_2. \end{aligned} \tag{9}$$

(8) and (9) fully characterize the dynamics of the the robot.

B. Continuous-Time State-Space Model

The state vector $x = [x_1, x_2, x_3, x_4]^T = [\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]^T$ is defined. From (8) and (9), the dynamics of the robot can be rewritten in state-space form

$$\dot{x} = \begin{pmatrix} x_2 \\ g_1(x, u) \\ x_4 \\ g_2(x, u) \end{pmatrix}. \tag{10}$$

Where

$$\begin{aligned} g_1(x, u) = &-\frac{k_1 \sin(x_1 - x_3) x_4^2 - u_1 + k_2 \cos(x_1)}{k_5 - k_6 \cos^2(x_1 - x_3)} \\ &- \frac{\cos(x_1 - x_3)(k_1 \sin(x_1 - x_3) x_2^2 + u_2 - k_3 \cos(x_3))}{-k_1 \cos^2(x_1 - x_3) + k_4}, \end{aligned} \tag{11}$$

and

$$\begin{aligned} g_2(x, u) = &\frac{k_9(k_1 \sin(x_1 - x_3) x_2^2 + u_2 - k_3 \cos(x_3))}{-k_7 \cos^2(x_1 - x_3) + k_8} \\ &+ \frac{\cos(x_1 - x_3)(k_1 \sin(x_1 - x_3) x_4^2 - u_1 + k_2 \cos(x_1))}{-k_1 \cos^2(x_1 - x_3) + k_4}, \end{aligned} \tag{12}$$

such that

$$\begin{gathered} k_1 = L_1 L_2 M_2 \\ k_2 = L_1 g (M_1 + M_2) \\ k_3 = L_2 M_2 g \\ k_4 = L_1 L_2 M_1 + L_1 L_2 M_2 \\ k_5 = {L_1}^2 M_1 + {L_1}^2 M_2 \\ k_6 = {L_1}^2 M_2 \\ k_7 = {L_2}^2 {M_2}^2 \\ k_8 = {L_2}^2 {M_2}^2 + M_1 {L_2}^2 M_2 \\ k_9 = (M_1 + M_2) \end{gathered} .$$

C. Optimal Control Formulation

The control objective is to move the robot from position x_a to position x_b while satisfying input and state-constraints. One can describe this as an optimal control problem by finding u that minimizing the some cost function J. The cost function considered in this report is

$$J = \int_{t_0}^{t_f} ((x - x_f)^T Q (x - x_f) + u^T R u) dt, \quad t_0 < t < t_f, \tag{13}$$

Where t_0 and t_f are the initial and final times respectively. $Q \in \mathbb{R}^{4\times4}$ is a semi-positive definite diagonal matrix. $R \in \mathbb{R}^{2\times2}$ is a positive definite diagonal matrix. The weighing of each of the two matrices allows for control over penalization for either input energy or failure of not reaching final state over-time.

We define the Cartesian coordinates of any point on the two link robot as $(\bar{X}, \bar{Y})$. The kinematic equations for any point located on the two-link robot are

$$\begin{pmatrix}\bar{X}\\ \bar{Y}\end{pmatrix} = \begin{pmatrix}a\cos(x_1) + b\cos(x_1+x_3)\\ a\sin(x_1) + b\sin(x_1+x_3)\end{pmatrix}, \quad \text{s.t. } 0 \le a \le L_1, \text{ s.t. } 0 \le b \le L_2. \quad (14)$$

The state constraints are defined by the region D where the obstacles preventing the robot from reaching the target location are located (as shown in Fig. 1) and by the speed limits on any of two joints. One must make sure that all the points on the robot (and not just the end-effector) do not collide with the any of the obstacles. Hence the state-constraints can be defined as

$$(\bar{X}, \bar{Y}) \notin D \forall a, b, \quad |x_2| \le v_{max,1}, \quad |x_4| \le v_{max,2}. \quad (15)$$

We define the set of all valid state vectors as $\bar{D}$. The input constraints allow for a limit on how much torque can be applied on each joint of the robot. Hence,

$$|u_i(t)| \le u_{m,i} \in \mathbb{R}^+. \quad (16)$$

Hence the optimal control problem formulation can be described as

$$u^*(t) = \arg\min_u J \quad \text{such that} \quad (10),(15),(16) \text{ are satisfied} \quad (17)$$

D. Discretisation of Continuous-Time Dynamics

The Forward Euler method is applied to do the discretisation of the state-space equation (1). Let Δt denote the sampling time. We define discretized state and input vectors x_d and u_d such that $x_d(k) = x(kT)$ and $u_d(k) = u(kT)$ for all $k \ge 0$. Hence the discrete-time state-space model can be described as

$$x_d(k+1) = x(k) + f(x(k), u(k))\Delta t \quad (18)$$

III. Model Predictive Control

Model Predictive Control (MPC) is a control technique that uses the discrete-time dynamical model of the system to predict and optimize the future behavior of a system over a set horizon. The control strategy is determined by minimizing the cost function across the prediction horizon. Only the first control action is put into practice once the best course of action for the current time step has been calculated. After that, the horizon is moved forward, and the procedure is repeated at the following time step with updated data and predictions. This concept is called receding horizon control [8]. MPC is especially useful for regulating input, output, and system internal state constraints. Infinite horizon MPC aims to find the optimal input trajectory of all future time based on the discrete time cost. Hence, the cost function J_∞ would be defined as

$$J_\infty = \sum_{k=k_0}^{\infty} (x(k) - x_f)^T Q (x(k) - x_f) + u^T R u, \quad (19)$$

such that all state and input constraints are satisfied. However, finding the optimal input trajectory for this cost function is infeasible due to the infinite sum [8]. There are different approaches in the MPC literature to approximate the infinite horizon input trajectory. Two of these approaches are utilized in this paper.

A. Approach A: Using a Long Horizon

One way to approximate the cost function J_∞ is to use a long enough horizon N such that

$$J_\infty \approx J_A = \sum_{k=k_0}^{k_0+N} (x(k) - x_f)^T Q (x(k) - x_f) + u^T R u. \quad (20)$$

The prediction horizon N is important as it helps shape control actions, optimize performance, and guarantee that the dynamical system has its objectives satisfied. Since it enables the controller to consider more future information, a larger horizon typically leads to greater control performance. But doing so increases processing complexity, which might not be practical for real-time applications. If N is small, the MPC focuses more on the immediate behavior of the system. Although this lowers computational complexity, it could result in a less precise behaviour forecast for the system, thereby compromising long-term performance. Hence a trade-off for N must be considered [9].

B. Approach B: Terminal Region State Constraint

One can reduce the prediction horizon by incorporating a end horizon such that x must be in the neighborhood of x_f at the end of the horizon neighborhood. This can be mathematically described as

$$J_B = \sum_{k=k_0}^{k_0+N} (x(k) - x_f)^T Q (x(k) - x_f) + u^T R u, \quad \text{s.t } |x(k_0+N) - x_f|^2 < \epsilon \quad (21)$$

Where $\epsilon \ge 0$. Terminal constraints are employed to ensure that the system reaches a final state that optimizes a certain cost function. This is crucial for applications where long-term performance is as important as immediate control actions. The controller can plan future actions as the horizon advances and maintain consistent performance by making sure that every segment of the control horizon satisfies the terminal requirement [10].

One downside of Approach B is the potential conflict between the terminal constraints, state, and input constraints. There may exist a case where there is no input trajectory u that allows the terminal constraint condition to be satisfied within the prediction. This may especially happen if the chosen value for ϵ is taken to be "too small". The smaller the value of ϵ, the

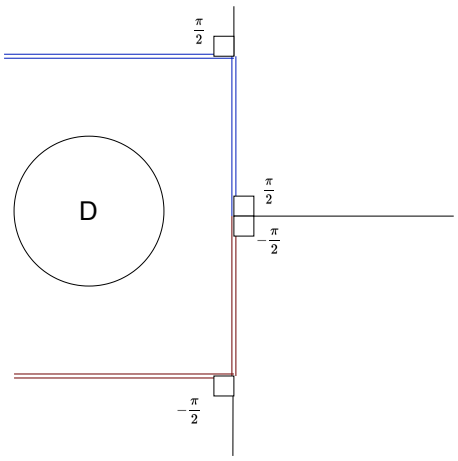


Fig. 2: An illustration showing the initial and final configurations of the robot. The starting configuration is drawn in blue. The final configuration is drawn in red.

more likely this constraint conflict is likely to occur. However, as ϵ is increased, one may need to increase the prediction horizon to obtain satisfactory control performance [10].

IV. Results and Discussion

A. Experimental Setup

The results presented in this paper are based on the following control objective. Consider a two-link arm robot with $L_1 = L_2 = 1$m, $M_1 = M_2 = 1$kg, and $g = 9.81$m/s^2 The robot is initially in the configuration of $\theta_1 = \pi/2$ and $\theta_2 = \pi/2$ with zero-initial velocity on all joints. The goal is to drive the robot to angular configuration $\theta_1 = -\pi/2$ and $\theta_2 = -\pi/2$ with zero final speed. Hence we can define

$$x_0 = \begin{pmatrix} \frac{\pi}{2} \\ 0 \\ \frac{\pi}{2} \\ 0 \end{pmatrix}, \quad x_f = \begin{pmatrix} \frac{-\pi}{2} \\ 0 \\ \frac{-\pi}{2} \\ 0 \end{pmatrix}.$$

There is a circular region D that is located between the initial and target position. This circular region is centered at $(0.75, 0)$m and has a radius of 0.4m. The maximum speed for each joint is limited to 5m/s. A maximum of $10N$ can be applied on each joint. Hence the state and input constraints can be defined as

$$\begin{gathered} (\bar{X} - 0.75)^2 + (\bar{Y})^2 \geq 0.4^2 \\ |x_2| \leq 5, |x_4| \leq 5 \\ |u_1| \leq 10, |u_2| \leq 10 \end{gathered} .$$

A high level overview of the example control objective is shown in Fig. 2. We define Q and R as

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, R = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}.$$

The sampling time is taken to be $\Delta t = 0.1$. The values of the N and ϵ are tuned throughout the experiments.

B. Cost Function Optimizer

We utilize the interior-point method (IPM) presented in [11] to optimize the cost functions J_A and J_B. Interior point methods (IPMs) are a widely used technique for solving various nonlinear optimization problems. IPMs work well for extremely large-scale issues as they have low-degree polynomial worst-case complexity and offer solutions in nearly constant iterations that do not greatly depend on the problem's size [12]. These features make them perfect for a high-dimensional nonlinear optimization such as the one discussed in this report.

To ensure that the fmincon optimizer works within realistic times, the author had to identify a successful yet sub-optimal trajectory as an initial guess for the optimal input trajectory.

C. Approach A Results and Discussion

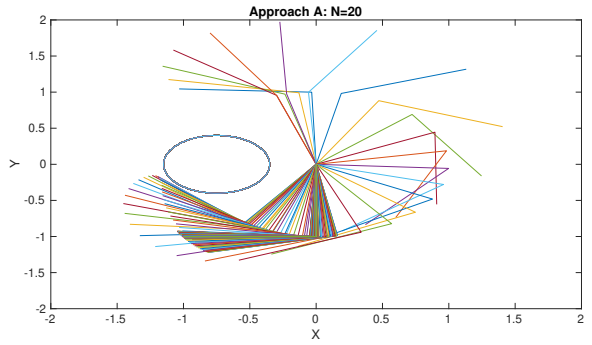
The main results for Approach A are presented in Fig. 3 and Fig. 4. Fig. 3 shows the results using Approach A and $N = 20$, whereas Fig. 4 shows the results using Approach A and $N = 30$. As one can see both prediction horizon parameters were able to satisfy the objective of moving the arm from x_0 to x_f without violating either the state or input constraints. However, we do notice that there is less transient overshoot in the $N = 30$ case and the robot is driven to x_f quicker in the $N = 30$. This implies as an N increases the control law is better optimized for long-term performance. which means there is likely to be overshooting as the cost function optimizer can predict future states better. Mathematically, this makes sense as it relates to the approximation presented in (20); as N increases, this approximation is more accurate to the infinite-horizon MPC case. The author found that below $N = 20$ while using Approach A, some issues may arise and the control objective is not satisfied.

D. Approach B Results and Discussion

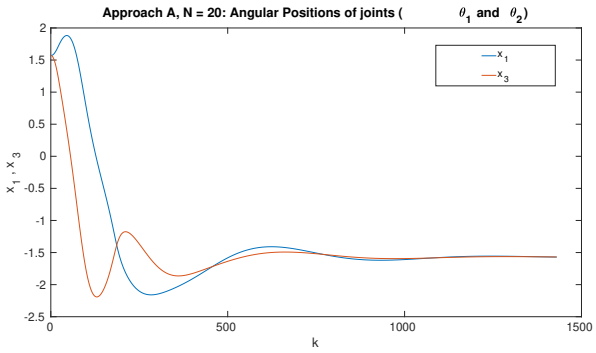
The main results for Approach B are presented in Fig. 5. We consider $\epsilon = 0.1$ and $N = 15$. As one can see the control objective is satisfied without violating either the state or input constraints. Furthermore, Approach B could work with a smaller prediction horizon compared to Approach A, which shows Approach B has the potential to be used in situations were lower computational complexity is needed.

E. Failure Cases

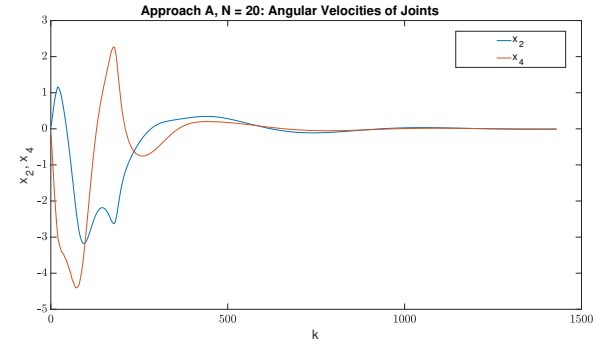
1) Prediction Horizon is too Small: Fig. 6 shows the results of the simulation using Approach A and $N = 5$. The state-constraints were neglected to put an emphasis on the failure of the control objective. As one can see, the arm never converges to x_f. Instead it continues to oscillate around in the neighbourhood around x_f. There is a relatively large overshoot in the transient stages. This result is expected as the cost function optimizer has a poor understanding of what the predicted states due to the small prediction horizon.



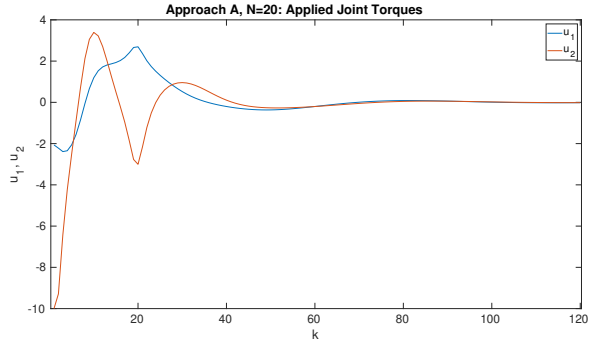
(a) The trajectory of the arm in the Cartesian space.



(b) The trajectory of the angular positions of the joints θ_1 and θ_2 (x_1 and x_3).

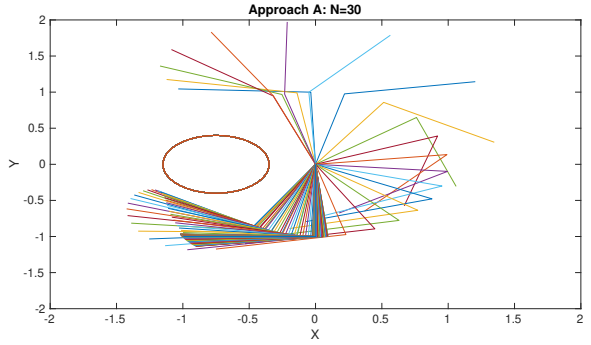


(c) The trajectory of the angular velocities of the joints $\dot{\theta_1}$ and $\dot{\theta_2}$ (x_2 and x_4).

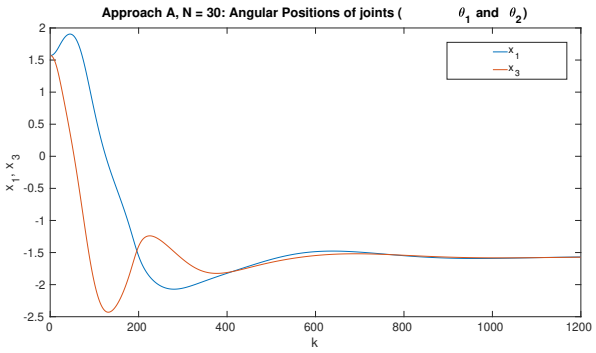


(d) The trajectory of the applied torques on the joints u_1 and u_2

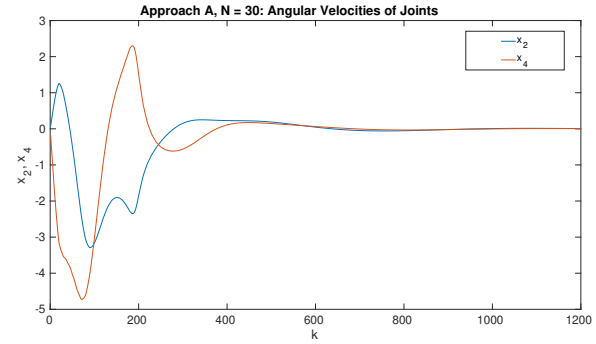
Fig. 3: The results obtained utilizing Approach A and $N = 20$.



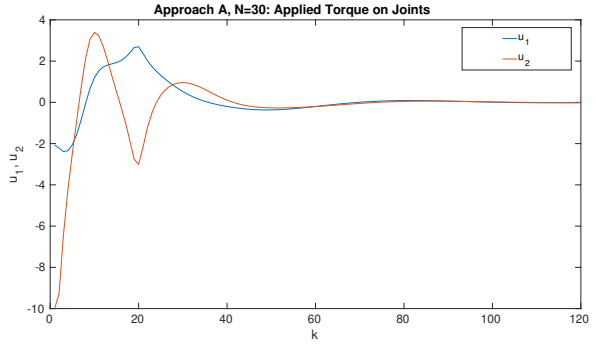
(a) The trajectory of the arm in the Cartesian space.



(b) The trajectory of the angular positions of the joints θ_1 and θ_2 (x_1 and x_3).

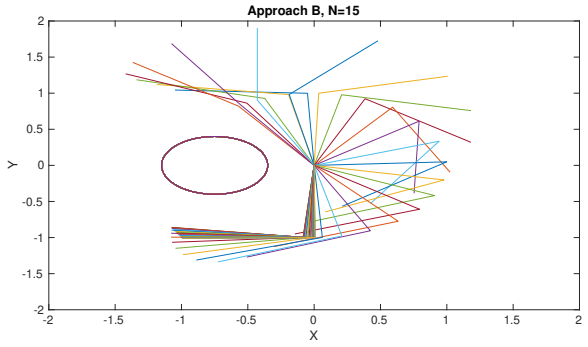


(c) The trajectory of the angular velocities of the joints $\dot{\theta_1}$ and $\dot{\theta_2}$ (x_2 and x_4).

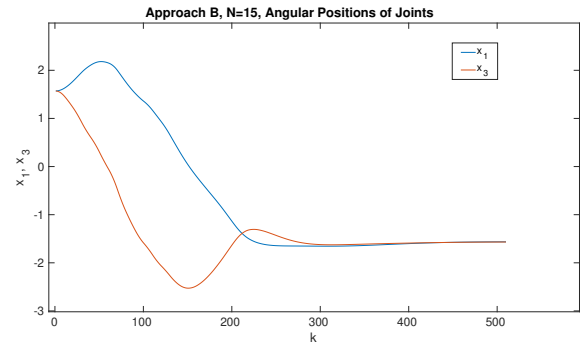


(d) The trajectory of the applied torques on the joints u_1 and u_2

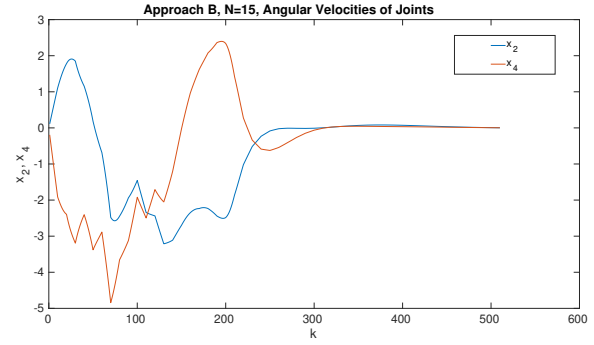
Fig. 4: The results obtained utilizing Approach A and $N = 30$.



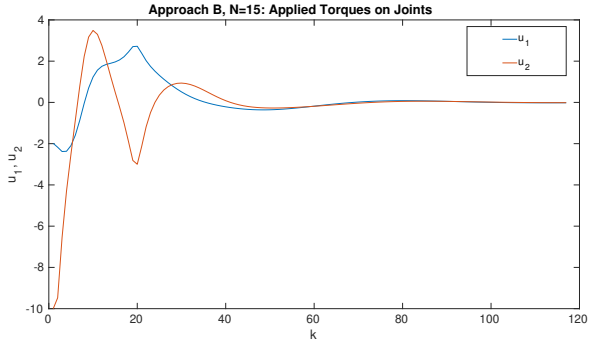
(a) The trajectory of the arm in the Cartesian space.



(b) The trajectory of the angular positions of the joints θ_1 and θ_2 (x_1 and x_3).

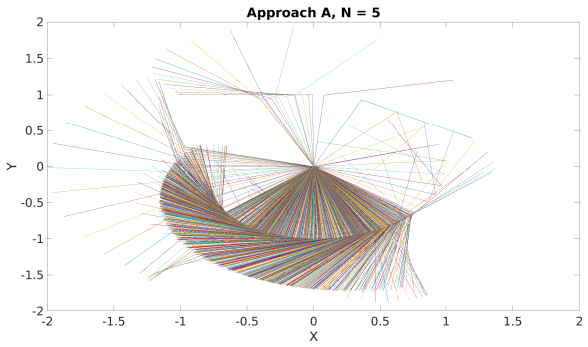


(c) The trajectory of the angular velocities of the joints $\dot{\theta_1}$ and $\dot{\theta_2}$ (x_2 and x_4).

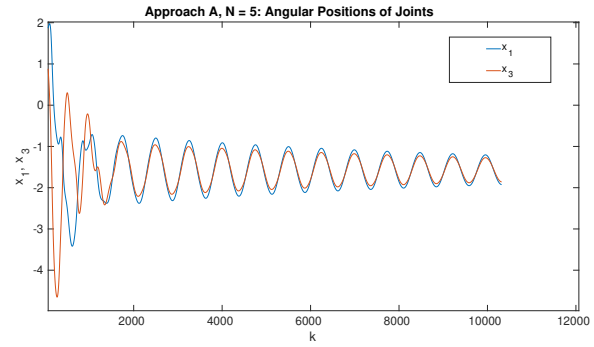


(d) The trajectory of the applied torques on the joints u_1 and u_2

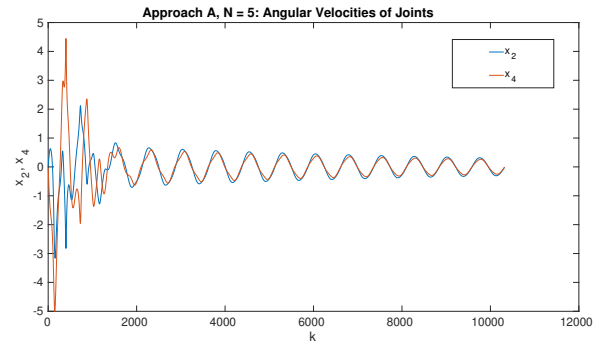
Fig. 5: The results obtained utilizing Approach B, $N = 15$, and $\epsilon = 0.1$



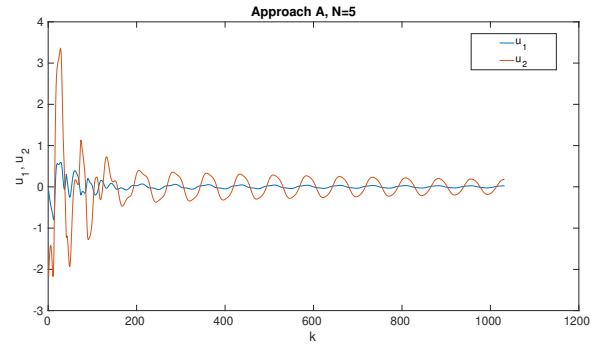
(a) The trajectory of the arm in the Cartesian space.



(b) The trajectory of the angular positions of the joints θ_1 and θ_2 (x_1 and x_3).



(c) The trajectory of the angular velocities of the joints $\dot{\theta_1}$ and $\dot{\theta_2}$ (x_2 and x_4).



(d) The trajectory of the applied torques on the joints u_1 and u_2

Fig. 6: The results obtained utilizing Approach A and $N = 5$.

2) Potential Infeasibility issues in Approach B: one issue that can arise in Approach B is if ϵ is set to a very small positive value and N is also small. For example, the author set $\epsilon = 0.01$ and $N = 10$ for one of the experiments and the optimizer failed to find a feasible solution. This is expected as the arm can't be driven to the defined neighbourhood around x_f within small prediction horizons. One needs to either to increase ϵ or increase N (or both) to ensure that the defined neighbourhood is large enough such that the arm can be driven to it within the horizon time.

F. Further Comments on Cost Function

1) Comments on Cost Function Matrix Parameters: Generally as established in the literature, in a quadratic cost function such as the one utilized in this paper, Q is positive semi-definite diagonal matrix and R is positive definite diagonal matrix [13]. Although one one pair of possible Q and R have been presented so far in this report, the author generally recommends the following Q and R

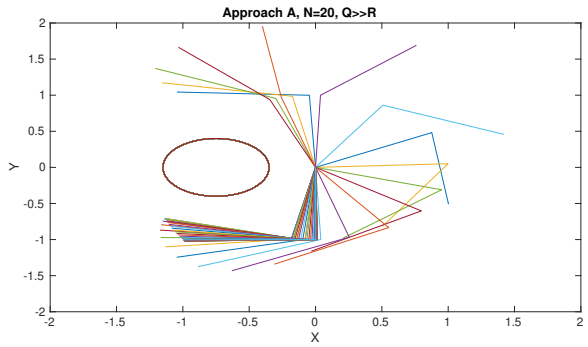
$$Q = \begin{pmatrix} k_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & k_2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, R = k_3 \begin{pmatrix} M_1 + M_2 & 0 \\ 0 & M_2 \end{pmatrix}. \quad (22)$$

Where k_1, k_2, and k_3 are positive number tunable parameters. The second and fourth diagonal entries of Q are zero as we do not care about the speed of the joints being close to zero throughout the trajectory. We only care about zero speed at the end of the journey. Furthermore, the first diagonal entry of R is greater than the second diagonal entry as we would want to put more cost weight on applying torque on the "shoulder" joint as it would effectively move the entire arm around and expend more energy.

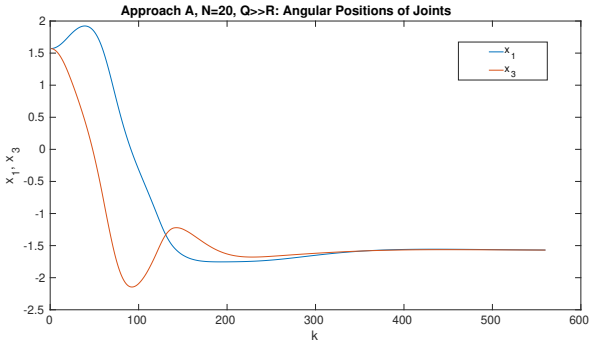
To highlight the effects of parameters k_1, k_2, k_3, the author modified Q and R such that $k_1 = k_2 = 15$ and $k_3 = 1$. This would put more weight on having the arm trying to reach x_f quicker and putting less weight on penalizing energy consumption. The results for Approach A using $N = 20$ with these modified cost function parameters are presented in Fig. 7. One can see there is a more of a "bang-bang" control approach in the transient stages of the trajectory, which is expected due to the above reason.

V. Future Work

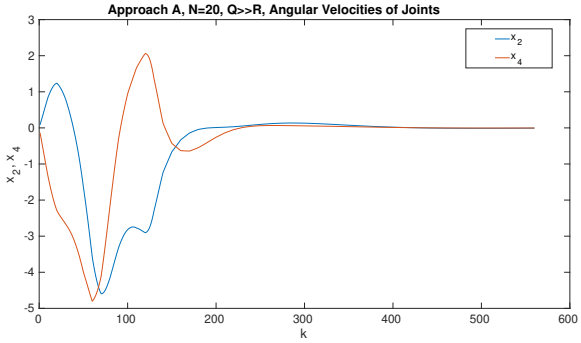
There are a few directions this work could be built on. Firstly, the work could be replicated on a real robot (or at least a simulated robot with the parameters of a real robot). Furthermore, alternative MPC approaches could be explored. For example, the author hypothesizes that one could design a mimimum-time cost function by making the prediction horizon N to be an optimization value. Hence, each iteration of the MPC framework would have a different prediction horizon. Furthermore, the obstacles were assumed to be static in this paper. Hence, the algorithm presented in this paper may need to be adjusted to moving obstacles. Finally, there is a question



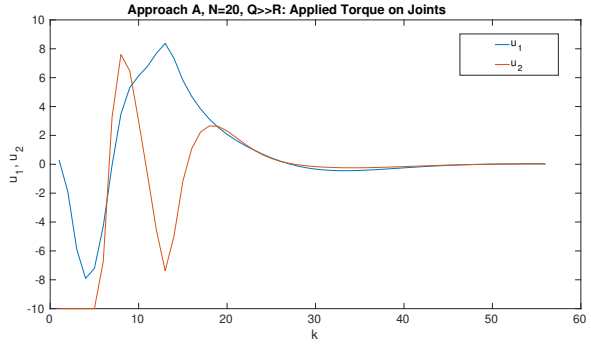
(a) The trajectory of the arm in the Cartesian space.



(b) The trajectory of the angular positions of the joints θ_1 and θ_2 (x_1 and x_3).



(c) The trajectory of the angular velocities of the joints $\dot{\theta_1}$ and $\dot{\theta_2}$ (x_2 and x_4).



(d) The trajectory of the applied torques on the joints u_1 and u_2

Fig. 7: The results obtained utilizing Approach A, $Q >> R$, and $N = 20$.

on the robustness of this algorithm to non-ideal dynamics that have not been modeled such as friction.

One fundamental flaw in the work presented in this paper is the need to have a reasonable initial guess for the optimal control law due to the non-convex nature of the dynamics which causes the optimizer to find local minima as solutions instead of the global minimum. This is not realistic in real scenarios. A more detailed research needs to be done on how to solve this issue. This thesis has provided a good starting point [14].

VI. Conclusion

This paper explored the use of nonlinear model predictive control (MPC) on a two-link rotating arm-robot using two separate approaches. MPC revolves around applying a closed-loop control law based on predicted future states. First, the dynamics of the two-link arm robot were derived and the optimal control law was formulated. The first MPC approach this paper explored was using a long prediction horizon to approximate infinite-horizon MPC. The second approach utilized terminal-state constraints to guide the cost function in lower prediction horizon scenarios. Each approach had its advantages and disadvantages. In the future, the author plans on tuning the algorithm in this work for more realistic scenarios so it can be applied on a real robot.

References

[1] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, "Recent trends in task and motion planning for robotics: A survey," *ACM Comput. Surv.*, vol. 55, jul 2023.

[2] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, *Path Planning and Trajectory Planning Algorithms: A General Overview*, pp. 3–27. Cham: Springer International Publishing, 2015.

[3] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, 2023.

[4] B. I. Kazem, A. I. Mahdi, and A. T. Oudah, "Motion planning for a robot arm by using genetic algorithm," 2008.

[5] M. Massaro, S. Lovato, M. Bottin, and G. Rosati, "An optimal control approach to the minimum-time trajectory planning of robotic manipulators," *Robotics*, vol. 12, no. 3, 2023.

[6] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 174–179, 2017.

[7] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. USA: Addison-Wesley Longman Publishing Co., Inc., 2nd ed., 1989.

[8] E. F. Camacho and C. Bordons, *Introduction to Model Predictive Control*, pp. 1–11. London: Springer London, 2007.

[9] E. F. Camacho and C. Bordons, *Nonlinear Model Predictive Control*, pp. 249–288. London: Springer London, 2007.

[10] E. F. Camacho and C. Bordons, *Constrained Model Predictive Control*, pp. 177–216. London: Springer London, 2007.

[11] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Mathematical Programming*, vol. 89, pp. 149–185, Nov. 2000.

[12] "Choosing the algorithm." https://www.mathworks.com/help/optim/ug/choosing-the-algorithm.htmlbsbwxm7. Accessed: 2024-04-27.

[13] A. Shaiju and I. R. Petersen, "Formulas for discrete time lqr, lqg, leqg and minimax lqg optimal control problems," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 8773–8778, 2008. 17th IFAC World Congress.

[14] K. Bergman, *Exploiting Direct Optimal Control for Motion Planning in Unstructured Environments*. Ph.d. thesis, Linköping University, Linköping, Sweden, 2021.