Nonlinear MPC for Robotic Arm Path Planning and Control ECH-267 Final Project Report

Jonathan Dorsey *Member: No Sleep Club: est. 2017*

***Abstract*—The objective of this project is to implement a simulated optimal Path Planner using Model Predictive Control (MPC) to plan and control the behavior of a 2 degree of freedom (2DOF) robotic arm. The responsibility of the MPC planner will be to generate the 'optimal' path and to drive the arm from its current position to the next. The main challenges faced in completing this project consist of solving for the nonlinear equations of motion (as well as any required forward/inverse kinematics) of the robotic arm as well as formulating and solving the MPC controller, at each timestep, both control and planning using the CasADi optimization framework, to test the scenario of simulated real-time performance.**

***Index Terms*—Model Predictive Control, Robot Arm, Lagrange Equations, Path Planning, Obsticle Avoidance, DH Parameters, Trajectory Generation.**

I. Introduction

THE world of robotics is full of constraints and demands that humans handle naturally and with ease on a daily basis. However, for robotic systems, planning motion or tasks around constraints, limitations, and desired outcomes is a non-trivial task which demands solutions techniques (control and planning) that capable of integrating both hard and soft constraints. Such constraints could manifest across many domains including system performance, physical mobility, design factors, actuation and hardware limitations, to application or task specific processes. One of the many challenges faced in the control design for robotic manipulators conerns how to take all of these constraints into consideration while still performing the assigned tasks.

Model Predictive Control (MPC) is an optimal control methodology which solves the a given optimal control problem (OCP) in a receding fashion, over a finite horizon. While these controllers are far more sophisticated than standard classical or modern control strategies, the increased complexity and computation can be applied to a wider class of control problems.

One such class of systems are Multiple-Input-Multiple-Output (MIMO), where classical controllers (PID... etc.) become much more involved and less straightforward to utilize effectively. This problem is only exacerbated when the system is nonlinear, which might require techiniques like linearization, feedback-linearization, or gain-scheduling, to effectively control the system using classical or modern control methods.

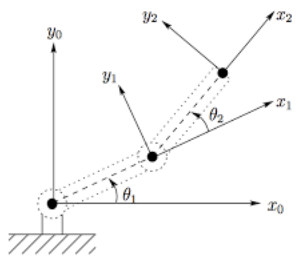
MPC, on the other hand, is a versitile methodology which naturally extends to MIMO systems and seamlessly integrates constraint handling. This flexibility is a property of the underlying structure of the optimal control problem. However, unlike other optimal control solutions such as Linear Quadratic Regulators (LQR), MPC benefits from feedback and thereby increases robustness for modeling or stoichastic uncertainties.

While still fundamentally a control strategy, one of the unique benefits of MPC is that (over its prediction horizon) it provides the optimal inputs to achieve a desired objective. Since a model of the system is required for MPC to work, we can propigate the predicted state(s) of the system by using the current initial condition, and the optimal control inputs. This in turn produces the optimal state trajectory, at the given time step over the prediction horizon.

The objective of this project is to use MPC to plan and control the motion of a 2D robotic manipulator. Additionally, I want to compare and contrast the tracking performance of different controllers to over the

II. Problem Statement

Problem Statement Here



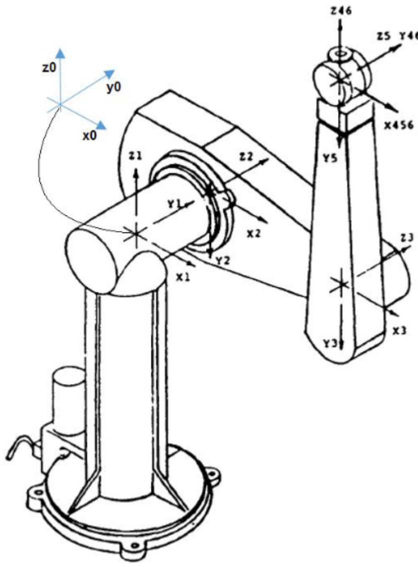
III. Background

To discuss the subject of this project in any detail, it is requisite to cover the basic terminology, constructs, and definitions which will be used throughout the remainder of this paper.

A. Robotic Manipulators

The worlds of robotics and industrial automation are a extremely broad. For the purposes of this project, I plan to reduce the scope of this field to the study of a 2-Link Planar robotic manipulator, subject to gravity.

In general robots typically possess prismatic (linear) or revolute (rotational) joints. The PUMA 560 robot (INLCUDE FIGURE), is a typical example of an industry standard manipulator, with 6 degrees of freedom (DOF).



For this project, the size and complexity even the 3 DOF dynamics required to control the first three links of the PUMA 560 are too extensive to be derived or used in any practical fashion. As this is the case, the simplier case for the planar two link robot will be used as the test platform for this project.

In general,

B. Trajectory vs Path Generation

One of the details that often gets overlooked in the difference between a path and a trajectory. There is, however, a significant and meaningful distinction the two. Paths are independ of time and only consists of a sequence of positions which are meaningful in some way. Trajectories, on the other hand, are dependent on time. This means that for every position in a given trajectory, there is an associated time. This terminology dictates that velocity and acceleration sequences desribe trajectories since both quantities are time dependent. It should be noted that paths can be converted into trajectories by restricting that positions on the path are tracked according to some dependence on time.

In the context of this paper, **path generation** will suffice, since I am only interested in making sure that robotic arm is driven to desired pose (position and orientation) regardless of the time it takes to do so. This is nice simplification to studying and solving trajectory tracking problems and has the nice analytical benefit of making the system dynamics autonomous.

C. Optimal Path Generation

One interest fact about path planning is that there are infinitely many ways to traverse to two fixed points in continuous space. Therefore one of the engineering challenges is to generate a path with desirable characteristics, for the system being studied. In many cases, it is desirable to obtain the path which defines the shortest distance or time between two points, achievable for a given system.

In a broader sense, this means finding the optimal path according to some measure of fitness. In the context of robot arm manipulation, the optimal path is that which will direct the robot from an initial pose to a final pose without violating system or path constraints and could even be extended to avoiding obsticles which would otherwise collide with the robot manipulator.

D. Model Predictive Control

As previously illuded to, MPC is an optimization based controller that solves for the

E. Path Planner

F. Articulated Robotic Systems

1) Equations of Motion:

2) Denavit-Hartenberg Paramters:

The Denavit-Hartenberg (DH) parameters are an important tool in analyzing the geometry of any given robot as well as in the formulation of joint transformations which enable a concise and universal means of deriving important quanties for kinematic and dynamic analysis of the robot.

The purpose of DH parameters is to standardize the description of geometry of robots into a universal parameterization such that arbitary construction of different robots can all be described concisely in a single and intuitive notation.

The DH Parameters are defined to be...

$$\begin{aligned} a_i &= \text{ the distance from } \hat{Z}_i \text{ to } \hat{Z}_{i+1} \text{ measured along } \hat{X}_i \\ \alpha_i &= \text{ the angle from } \hat{Z}_i \text{ to } \hat{Z}_{i+1} \text{ measured about } \hat{X}_i \\ d_i &= \text{ the distance from } \hat{X}_{i-1} \text{ to } \hat{X}_i \text{ measured along } \hat{Z}_i\text{; and} \\ \theta_i &= \text{ the angle from } \hat{X}_{i-1} \text{ to } \hat{X}_i \text{ measured about } \hat{Z}_i \end{aligned}$$

The primary usage of DH Parameters is the generation of transformations to and from joint frame origins. Before this can be achieved, there needs to exist the notion of a general transformation which captures both position and orientation (e.g. pose) from one frame to another. By coupling position vectors to the origin of each joint frame and the corresponding rotation matrix, we arrive at the following **homogeneous transforms**.

$$\begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{{}^{A}_{B}R} & {}^{A}P_{Borg} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix}$$

Which can be further simplified to...

$$P = {}_{B}^{A}T^{B}P$$

Since the DH Parameters provide a universal notation for describing the the position and orientation of a robot, it is natural to want to express the homogeneous transformation of each coordinate frame using this parameterization. For a single transformation from a frame $\{i\}$ to frame $\{i-1\}$, we can use the homogeneous transformation for explicit rotations about a joint axis or specific translations along a joint axis as dictated by the DH parameters.The matrices R and D represent the homogeneous transformation for rotation and translations repsectively, with the subscript of each providing the axis upon which the operation should be performed.

$${}_{i}^{i-1}T = R_X\left(\alpha_{i-1}\right) D_X\left(a_{i-1}\right) R_Z\left(\theta_i\right) D_Z\left(d_i\right)$$

By using the universal, DH paramterization with homogeneous transformation we can now describe the relationships to or from any joint of the robot to any other joint of the robot. This is a powerful concept which facilitates the analysis of any robot whose joints are based on revolute or prismatic members.

3) Forward Kinematics:

IV. Literature Review

Literature review goes here

V. Project Methodology

A. CasADi Software

B.

C.

D.

VI. MPC Formulation

The mathematical basis for MPC stems from the standard optimal control problem. However, with a few slight adjustments and relaxations, the formulation for MPC can be

A. MPC Assumptions

1) Internal dynamic model of system exists
2) Finite (and receding) prediction horizion
3) Piecewise constant input at each time-step

After applying these restrictions to the standard optimal control problem, the mathematical formulation for standard MPC reduces to the following expression.

$$\begin{aligned} \min_{n} J_x\left(\mathbf{x}_0, \mathbf{u}\right) &= \sum_{k=0}^{N-1} \ell\left(\mathbf{x_v}(k), \mathbf{u}(k)\right) \\ \text{s.t: } &: \mathbf{x_n}(k+1) = \mathbf{f}\left(\mathbf{x_n}(k), \mathbf{u}(k)\right) \\ &\mathbf{x_u}(0) = \mathbf{x}_0, \\ &\mathbf{u}(k) \in U, \forall k \in [0, N-1] \\ &\mathbf{x_u}(k) \in X, \forall k \in [0, N] \end{aligned} \tag{1}$$

In this formulation, the cost function is only propagated to the end of the prediction horizon N, from the current initial state of the system. The first equality constraint restricts future states of the system to be feasible with respect to the system model. The second equality constraints requires the initial state of the solver to be the current initial state of the system, while the final constraints require the inputs and states determined by the solver to be valid elements of feasible input and state sets respectively.

The particular flavor of MPC used in this project utilized the following quadratic stage cost.

$$\ell\left(\mathbf{x_v}(k), \mathbf{u}(k)\right) = (x - x_d)Q(x - x_d) + (u - u_d)R(u - u_d) \tag{2}$$

This parameterization stage cost can be further simplified using norms.

$$\ell\left(\mathbf{x_v}(k), \mathbf{u}(k)\right) = \|x - x_d\|_Q^2 + \|u - u_d\|_R^2 \tag{3}$$

The quadratic nature of this cost function is eligant in how simple and intuitive it is comprehend as as well how simple it is to tune using the matrices Q and R.

It should be further noted that this projet will be using **Multiple Shooting** formulation of MPC as this approach has shown to produce better results and obtained solutions far faster than the naive single shooting approach.

VII. Path Planner

An addition benefit to using multiple shooting is that since the state predictions over the time horizon are included as decision variables along with the input sequence, the very produce of solving for the optimal inputs (over the prediction horizon) also yields the predicted states without necessitating forward propigating the optimal input and current state through the system model.

Because of this fact, at any given instance, the MPC controller not only yields the optimal inputs, but specifies the optimal state trajectory over the prediction horizion.

By leveraging this fact, we can fit a cubic spline through those discrete state predictions. Once calculated, this spline can be used to as a refernce path to calculate reference joint positions for a controller to track.

Naturally, as the prediction horizon used for MPC probably does not span the entire path required to drive the robot from its initial pose to its desired pose, the reference path will be need to be recomputed occationally, if the predictions from the model deviate from the actual measurements of the system (or state estimator outputs).

VIII. Dynamics Model

For this project, the model of the robot will only include the dynamics of the physical system and not the dynamics of the actuators. However, it would be trivial to include these effects for a more realistic model of how the system operates, but for the purposes of simplicity, we will assume that the joint motors have perfect torque control.

A further simplification to modeling this system is that we assume massless links and that each point can be modeled as a point mass. Under these assumptions, we can ignore contributions by moments of inertia. This simplifies the process of deriving equations of motion.

The method to derive the dynamics of the modeled system uses the **Euler-Lagrange Equations**. This is variational method which uses the kinetic and potential energies of the system to derive the dynamics of the robot. While equivalent to the Newton-Euler Equations, Lagrange Equations do not require the computing of accelerations and typically simplify the derivation of

$$k_i = \frac{1}{2} m_i v_{C_i}^T v_{C_i} + \frac{1}{2}^i \omega_i^T C_i I_i^i \omega_i \tag{4}$$

$$u_i = -m_i^0 g^{T0} P_{C_i} + u_{ref_i} \tag{5}$$

$$k = \sum_{i=1}^{n} k_i \tag{6}$$

$$u = \sum_{i=1}^{n} u_i \tag{7}$$

$$L = k - u \tag{8}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{9}$$

A. Nonlinear Model

In general, the serially articulated robots with revolute joints will produce equations of motion (EOM) that take the following form.

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\dot{\Theta}) \tag{10}$$

In this formulation, M is the mass matrix which models the effects of mass and moments of inertia which are related to the angular accelerations of the joints. The V vector models the centrifugal and Coriolis forces which are typically functions of velocities, while the G vector models the effects of gravity. Finally, the F vector is added to explictly include the effects of friction/damping.

$$M(\theta) = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \tag{11}$$

Where

$$\begin{aligned} m_{11} &= m_1 L_1^2 + m_2\left(L_1^2 + 2L_1L_2\cos(\theta_1) + L_2^2\right) + \varepsilon \\ m_{12} &= m_2\left(L_1L_2\cos(\theta_2) + L_2^2\right) \\ m_{21} &= m_2\left(L_1L_2\cos(\theta_2) + L_2^2\right) \\ m_{22} &= m_2 L_2^2 + \varepsilon \end{aligned}$$

$$V(\theta, \dot{\theta}) = \begin{bmatrix} -m_2L_1L_2\sin(\theta_2)\left(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2\right) \\ m_2L_1L_2\dot{\theta}_1^2\sin(\theta_2) \end{bmatrix} \tag{12}$$

$$G(\theta) = \begin{bmatrix} (m_1 + m_2) L_1 g\cos(\theta_1) + m_2 g_2\cos(\theta_1 + \theta_2) \\ m_2 g L_2\cos(\theta_1 + \theta_2) \end{bmatrix} \tag{13}$$

$$F(\dot{\theta}) = \begin{bmatrix} c_f \cdot \dot{\theta}_1 \\ c_f \cdot \dot{\theta}_2 \end{bmatrix} \tag{14}$$

It should be noted for simulation purposes, the equations of motion need to express τ as the input to the robot and the angular accelerations $\ddot{\Theta}$ as the outputs.

$$\ddot{\theta} = \mathrm{inv}(M)[\tau - V - G - F] \tag{15}$$

However, expressing the governing equations in this form result in numerical instabilties during simulation which arise from inverting the mass matrix $M(\Theta)$. Since the mass matrix is function if the angular positions Θ, certain angular positions will cause the mass matrix to near a singularity, when inverted, at which the numerical values explode until they become **INF** or **NAN**. To avoid this, the value of $varepsilon$ can be added to the diagonal terms of M to garuntee that the system will never become singular, during inversion.

With equation (EQN NUMBER), we can now write the equivalent nonlinear state space for the system. By defining the states such that, $x_1 = \theta_1$, $x_2 = \theta_2$, $x_3 = \dot{\theta}_1$, and $x_4 = \dot{\theta}_2$, we can

$$\begin{aligned} \dot{x}_1 &= f_1(x, \dot{x}, u) \\ \dot{x}_2 &= f_2(x, \dot{x}, u) \\ \dot{x}_3 &= f_3(x, \dot{x}, u) \\ \dot{x}_4 &= f_4(x, \dot{x}, u) \end{aligned} \tag{16}$$

Such that

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \ddot{\theta}_1 \\ \dot{x}_4 &= \ddot{\theta}_2 \end{aligned} \tag{17}$$

Linearizing the system required computing the Jacobian of the nonlinear state space, and evaluating the resulting matrix at specified equilibrium points, such that around a small neighborhood the system will behavior linearly.

$$0.2\ \left(59200.0\,T_1 - 8000.0\,T_2 - 14800.0\,q_{\dot{1}} + 2000.0\,q_{\dot{2}} - 31390.0\,\cos(q_1\right.$$

IX. Tracking Controllers

A. *PID Control*

B. *LQR*

C. *MPC*

D. *H Infinity*

E. *Feedback Linerization*

X. Results

Results go here

XI. Discussion

Dicussion and points of note.

XII. Conclusion

The conclusion goes here.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank Mountain Dew and his matress for constant support and comfort.

[4] [6] [9] [1] [8] [7] [5] [2] [3] [10]

References

[1] B. Armstrong, O. Khatib, and J. Burdick. “The explicit dynamic model and inertial parameters of the PUMA 560 arm”. In: *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. Vol. 3. San Francisco, CA, USA: Institute of Electrical and Electronics Engineers, 1986, pp. 510–518. DOI: 10.1109/ROBOT.1986.1087644. URL: http://ieeexplore.ieee.org/document/1087644/ (visited on 03/05/2021).

[2] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. English. OCLC: 999637552. Cambridge: Cambridge University Press, 2017. ISBN: 9781139061759. URL: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4913401 (visited on 03/05/2021).

[3] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004. ISBN: 9780521833783.

[4] John J. Craig. *Introduction to robotics: mechanics and control*. eng. 3. ed., international ed. Pearson education international. OCLC: 249488662. Upper Saddle River, NJ: Pearson, Prentice Hall, 2005. ISBN: 9780131236295.

[5] Donald T. Greenwood. *Advanced dynamics*. Digitally printed 1st pbk. version. Cambridge ; New York: Cambridge University Press, 2006. ISBN: 9780521029933 9780521826129.

[6] Hassan K. Khalil. *Nonlinear systems*. 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2002. ISBN: 9780130673893.

[7] J. L. Meriam and L. G. Kraige. *Engineering mechanics*. 3rd ed. New York: Wiley, 1993. ISBN: 9780471592723 9780471592730.

[8] Katsuhiko Ogata. *Modern control engineering*. 5th ed. Prentice-Hall electrical engineering series. Instrumentation and controls series. Boston: Prentice-Hall, 2010. ISBN: 9780136156734.

[9] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model predictive control: theory, computation, and design*. eng. 2nd edition. OCLC: 1020170256. Madison, Wisconsin: Nob Hill Publishing, 2017. ISBN: 9780975937730.

[10] J.-J. E. Slotine and Weiping Li. *Applied nonlinear control*. Englewood Cliffs, N.J: Prentice Hall, 1991. ISBN: 9780130408907.

Jonathan Dorsey Biography text here.