Kwasi Debrah-Pinamang
ME 601 - Feedback Control of Autonomous Systems
Final Project Report
May 8th, 2024

Introduction

One of the most interesting, useful, and captivating problems in robotics research today is the topic of bipedal robots. Bipedal robots are favored for certain problems due to their mobility (especially in uneven terrain) and robustness in control [1]. Several companies, such as Boston Dynamics and Tesla, are currently in the process of research and development of bipedal robots in order to further the research of this class of robot and expand the range of problems that bipedal robots can be utilized for. For this project, I have decided to design, test, and simulate two control systems for bipedal robots.

Problem Statement

The goal of this project is to develop two control algorithms using two different methods, and then compare them to see which proves effective for this problem. The two control methods that I am utilizing here are two that we have investigated in class: Model Predictive Control (MPC) and Linear Quadratic Regulator (LQR). The bipedal robot model used in this project comes from a Mathworks Github repository, which further contains several resources for control system design of the bipedal robot system [2].

Algorithm Description

To begin, I will start by describing the Simulink model used for this project. Our main Simulink model holds two blocks that are integral to the model's operation: walking controller and walking robot. The walking controller block takes in the reference path and measurements regarding the robot's position and orientation, is what this project will mainly interface with, and it is mainly responsible for the control of the bipedal robot. This block mainly contains a block for the zero moment point, a block for step planning logic, and two blocks that run inverse kinematics on each of the legs in order to obtain angles. The angles of the legs are the main output of the control system, and are then fed into the walking robot controller. The walking robot block takes in the angles outputted from the controller block and utilizes the angles and the model within Simulink to generate the simulator. In addition, measurements regarding the

position and orientation of the robot are published from this block and fed back to the controller to form a feedback loop.

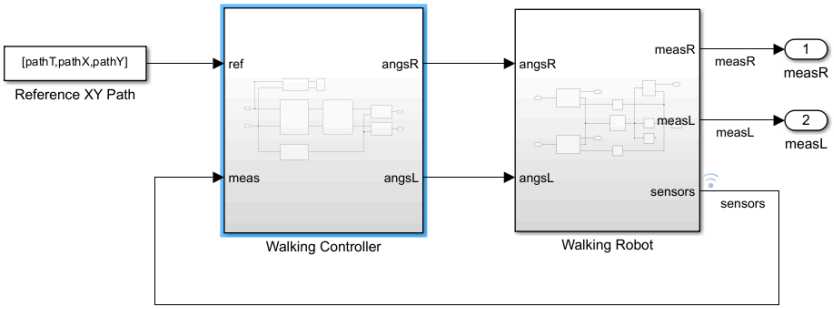

Figure 1: The Main Simulink Model for this System

Next, we will talk more about the Model Predictive Control algorithm. In order to build our model, we must first develop our state space model. In order to do this, we can think of our bipedal robot as an inverted pendulum [3]. With this image, we can develop our equations of motion. We can further implement the zero moment point into our equations of motion. For reference, the zero moment point can be thought of as the point directly below the center of mass of a bipedal robot. This is what we mainly want to control in our control system. With our equations of motion, we can now build our state-space model. Before we utilize Matlab's MPC function, we discretize our system due to the structure of our specific problem. With our discretized plant, we can now use the mpc() function with Matlab to build our controller. After defining the function, we can now set many of our system parameters. The system parameters that we set are the following:

- Prediction & control horizon - determines number of future steps for system behavior and control input computation
- Weights - specifies importance of objectives in control system
- Manipulated variable (MV) - variable(s) that can be modified to change system behavior
- Output variable (OV) - variable(s) that are being measured
- External Covariance Reset (ECR) - helps controller handle uncertainties/disturbances by adjusting how much belief the controller has in future predictions regarding system behavior

Each of these corresponds to specifics in our control system and impact the overall effectiveness of our model.

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x$$

$$p_x = \begin{bmatrix} 1 & 0 & -\frac{z_c}{g} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

Figure 2: Our state-space model

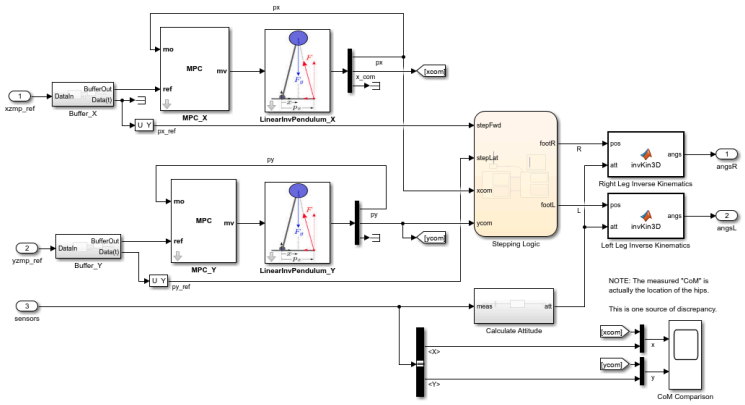

Figure 3: Our MPC Simulink Model

Now, we will dive more into the specific control algorithm, starting with the LQR controller. The Linear Quadratic Regulator control algorithm works by calculating an output based on the following matrices: A, B, C, D, P, Q, and R. A, B, C, and D are from the state space system, where these matrices can be thought of as the state matrix, input matrix, output matrix, and feedthrough matrix respectively [4]. Q and R are two control matrices that can be set to impact the system. Modifying these two matrices is what constitutes tuning the LQR control algorithm. Lastly, P is calculated using the Algebraic Riccati Equation (ARE) by using the previous matrices. In order to calculate u, our output, we use another matrix K by our error. K is calculated by using the matrices R, B, and P, while the error is simply the difference between our reference path and the input position. The LQR controller in this portion is used to control the acceleration of our robot.

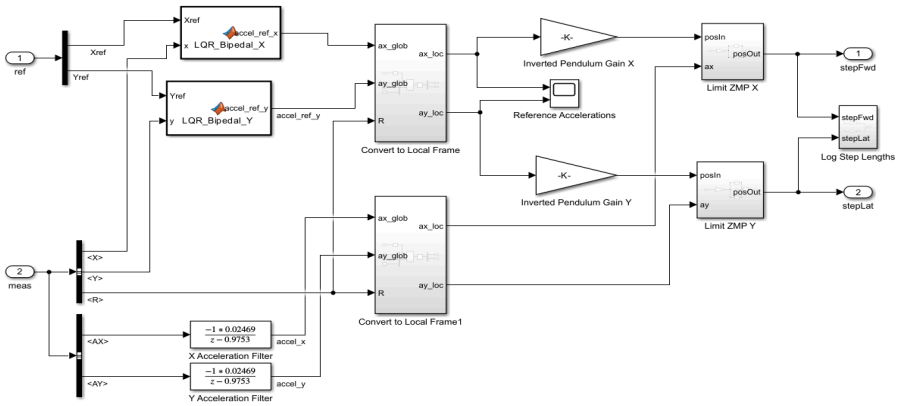

Figure 4: The Zero Moment Point Controller using LQR

Simulation Results

Now that we have discussed the aspects of control, we can talk about the results achieved via simulation. In order to visualize our results, we can use the Simscape Multibody add-on to simulate our model with our control algorithm. We can also graph out the trajectory and step pattern for our algorithm. The videos can be found in the zip file, but our graphical results are as follow:

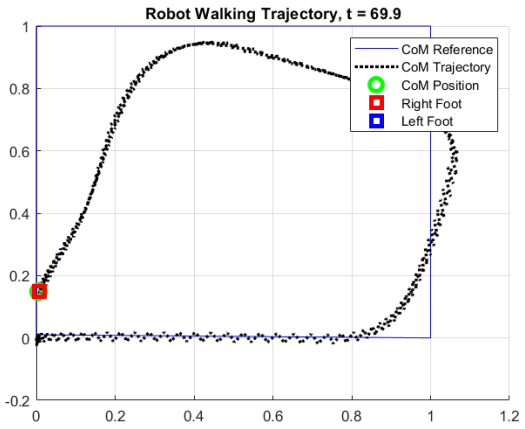

Figure 5: The trajectory of the LQR model

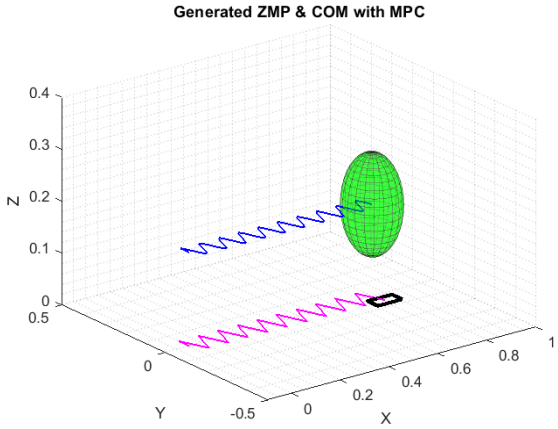

Figure 6: The walking pattern of the MPC model

Discussion

The results of these simulations can be seen in these two graphs. The left graph, the LQR, is meant to follow a square pattern, but instead fails to keep itself fully alongside the path. I am guessing with further tuning of the LQR algorithm, we would get a much more accurate

path-following algorithm. The right graph indicates the results of the MPC algorithm. It is a bit hard to see from the graph alone, but the walking pattern for the MPC algorithm is kind of awkward. It is easier to see in the simulation, but the amplitude of the waveform in the graph indicates this. Overall, both of these two algorithms provide strong results, and it is hard to definitively say which algorithm is better for this problem, though I think it is safe to say that each of these two algorithms is better suited to certain applications of this problem.

Conclusion

To conclude, I was able to achieve my goal of designing, testing, and simulating two of the optimal control methods that we discussed in class for this particular problem. I believe the results that I obtained were good, but like I alluded to a bit earlier, with more parameter tuning, both of these algorithms would likely provide better results. However, with how much trouble I encountered in this project, mainly with integrating my LQR algorithm and setting up the workspace, I am satisfied with my end results for this course's final project.

References

[1] T. Mikolajczyk *et al.*, "Recent Advances in Bipedal Walking Robots: Review of Gait, Drive, Sensors and Control Systems," *Sensors (Basel, Switzerland)*, vol. 22, no. 12, p. 4440, Jun. 2022, doi: https://doi.org/10.3390/s22124440.

[2] "mathworks/msra-walking-robot," *GitHub*, Apr. 22, 2024. https://github.com/mathworks/msra-walking-robot/tree/master (accessed May 09, 2024).

[3] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, doi: https://doi.org/10.1109/iros.2001.973365.

[4] Wikipedia Contributors, "State-space representation," *Wikipedia*, Oct. 14, 2019. https://en.wikipedia.org/wiki/State-space_representation