MAE 263C:

Circus Bot
Team: 10

Jonathan Bopp
Rong Jin

I. Overview

The problem of balancing an inverted pendulum is a classic controls problem. Many solutions have been developed which make this problem a veritable benchmark for comparing different control schemes. Often the inverted pendulum problem is based on a cart and pole model; a pendulum is hinged at one end to a linearly translating base. By applying a force, governed by a control law, to the cart over time it is possible to control the state of the pendulum. Manipulators inherently offer more capability than a cart by the nature of having additional degrees of freedom, however the manipulator and the pendulum are both non-linear systems. By introducing a manipulator in place of the cart additional complexities to the inverted pendulum problem must be addressed. Developing solutions to the swing up, balance, and overall control of the 3 linked system will serve as the core scientific content discussed in this paper.

A. Literature Review

There are multiple problems related to inverted pendulums, one solution which focuses on the balancing of the pendulum in the inverted orientation will be discussed. The method chosen to stabilize the inverted pendulum in a cart-pole scenario was a cascade controller, this is where the two outputs of the system are separated i.e.(X, ϕ). As shown in Fig.1 the respective block diagrams describing the transfer functions for the pendulum based on an input force F are used to describe the pendulum dynamics. From this representation the author then designs an appropriate controller which stabilizes each transfer function separately.

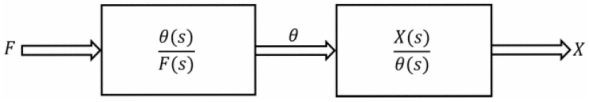


Fig. 1: Cascaded representation of pendulum dynamics [1]

By examining the the individual transfer functions and starting from the inner blocks a stabilizing controller was applied to each block. Ultimately the pendulum can be stabilized with two nested PD controllers. The controller developed here results in the control of both the pendulum base position X and pendulum angle ϕ.

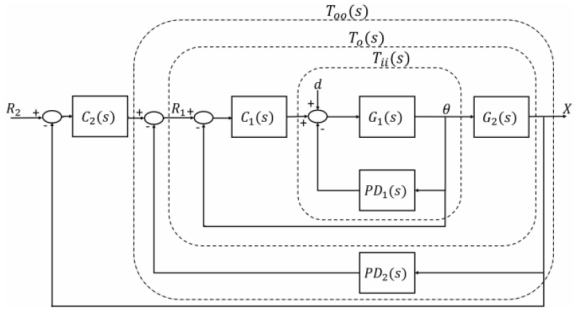


Fig. 2: Block diagram representation of inverted pendulum control using controller synthesis technique [1]

II. Objectives

Utilizing only a two link robotic manipulator to essentially control a three link system requires multiple controllers to be nested. At the core of this controlled system is the two link manipulator. An inverse dynamics controller is used to control the endeffector position as seen Fig.3. Along the periphery of this system lie the pendulum controllers. The pendulum controllers address

three problems swing-up, balance, and position control. The swing up controller utilizes an open-loop approach in order to impart enough energy into the pendulum so that the pendulum will rotate to the inverted orientation $\phi \approx 0$. The Inverted balance controller utilizes a full state feedback controller for the pendulum system. The position controller is within in the the full-state feedback controller for the pendulum as the states being controlled are both the endeffector position and the angle of the pendulum, as well as their derivatives, as measured from the positive Y-axis.

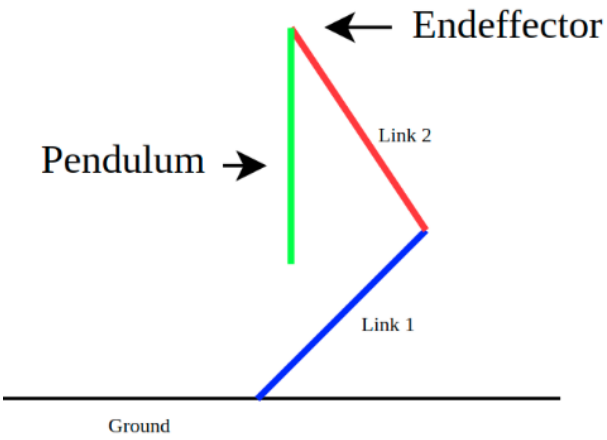


Fig. 3: Diagram labeling the parts of the 3 link system used in this project

Control of the two link manipulator is achieved using an inverse dynamics controller based in operational space. The controller for the manipulator requires the tuning of gains corresponding to the end effectors trajectory tracking in operational space. The gain matrices $K_p \& K_d$ correspond to the appropriate gains for the manipulator controller. For clarity the inverse dynamics controller is not comprised of solely the manipulator dynamics, in fact the inverse dynamics controller utilizes the full three link system. This style of compensation enables the manipulator to better compensate for the dynamics imposed onto the end of the manipulator by the moving pendulum. Later within the discussion section comments on the importance and influence of this compensation strategy will be addressed. The primary goal of the manipulator controller is to ensure trajectory tracking of the endeffector despite disturbances from uncompensated dynamics such that the inner pendulum controllers are able to operate successfully.

Phase Name	Controller Name	Condition
Swing Up	Sinusoidal Swing Up	$0.3 < \phi < -0.3$
Balance	Full-state feedback	$0.3 > \phi > -0.3$
N/A	Inverse Dynamics	$\forall \phi$

TABLE I: Table showing distinct phases controllers will be used as a function of pendulum angle ϕ

III. Methods

A. Inverse Dynamics

To place the pendulum full-state feedback controller outside the manipulator controller a crucial assumption must be stated. Assuming the endeffector of the manipulator is able to track a desired trajectory sufficiently with little error from un-compensated dynamics, then the inner pendulum system can be controlled with assumed decoupling from the manipulator dynamics. That is to say that the pendulum controller will not control for any other dynamics besides the pendulum. The dynamics of the pendulum can be stabilized assuming an input force is applied horizontally at the base of the pendulum. The control signal from the full-state feedback controller is routed to command the velocity of the endeffector as an input to the inverse dynamics controller. For this assumption to be held true the controller for the manipulator must be sufficiently fast and stable such that the pendulum controller is able to stabilize the pendulum. For this reason the settling time of $T_s < 0.1$ for a step response of (0.5m) is the requirement with no more than 5% overshoot. In order to achieve this level of performance the dynamics of the full 3 link system must be compensated for in the inverse dynamics controller. Quantities with an underscore s relate to the full 3 link system, terms relating to only the manipulator have an underscore m.

$$B_s(q)\ddot{q} + n_s(q, \dot{q}) = \vec{u}_m \tag{1}$$

$$\vec{u}_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \vec{u}_s \tag{2}$$

$$\vec{u}_s = B_s(q)y_s + n_s(q, \dot{q}) \tag{3}$$

$$\vec{y}_s = \begin{bmatrix} \vec{y_m} \\ 0 \end{bmatrix} \tag{4}$$

$$y_m = J_A^{-1}(q)(\ddot{x}_d + K_d\dot{\tilde{x}} + K_p\tilde{x} - \dot{J}_A(q, \dot{q})\dot{q}) \tag{5}$$

By developing the inverse dynamics controller in this way it is possible to arrive at a the double integrator

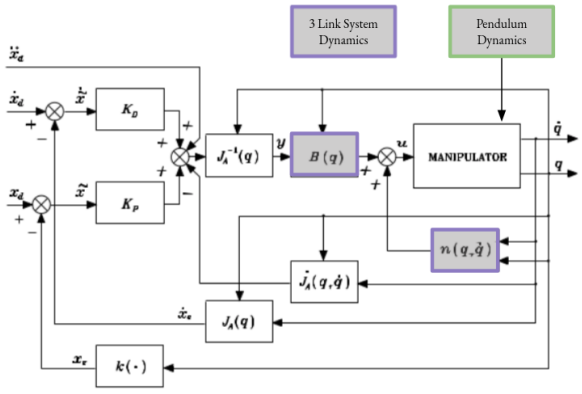


Fig. 4: Block diagram representation of the inverse dynamics controller with full system compensation

relationship for the first and second joints even though the system is comprised of three links. Eq.3 compensates for all the link dynamics by incorporating the control law for the two actuated joints shown in equation 5. A block diagram representing the implementation of Eq.1 can be seen below in Fig.4. It should be noted that although the dynamics of the pendulum are modeled within the full system for visual clarity the pendulum dynamics are modeled in Fig.4 as a disturbance on the manipulator system.

The system dynamics were determined using Euler-Lagrange formulation of all parts of the system including motor inertia's, gear reductions, and link geometry. Friction at the joints is accounted for in the full system but not in the inverse dynamics controller. The friction values are all identical($F_v = 0.5\frac{n-m-s}{rad}$ for each joint as the goal of this simulation is to account for and demonstrate control-ability based on arbitrary values rather than model a physically existing system. The dynamics were derived in such a way that the dynamics controller can be updated based on any arbitrary parameters specified for the system. Re-tuning would obviously be required.

B. Swing Up

In order to achieve a swing up controller that minimized displacement and final velocity of the pendulum a sinusoidal trajectory was selected. A sinusoidal trajectory with a frequency matched to the un-damped natural frequency of the pendulum would theoretically provide an energetic and quick response. However based on the estimated natural frequency of the pendulum the period would be longer than the time allotted. The open loop swing up controller relies on two parameters duration, and displacement. The time to complete the trajectory is set based on the properties of the pendulum length, the amplitude is then the only variable to tune such that the pendulum will be inverted with as little additional energy as possible. The swing up controller needs to reorient the pendulum to an inverted state in under a second. This time requirement is set because the operation should take relatively less time than the requirements for stabilizing the pendulum. The sinusoidal trajectory is visualized in Fig.5.

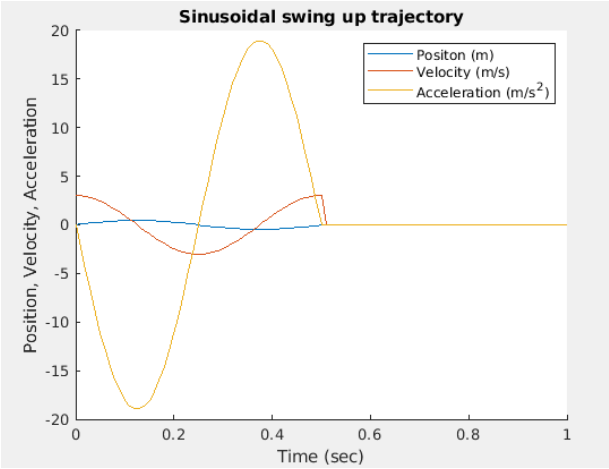


Fig. 5: Plot representing the sinusoidal trajectory commanded along the X-axis the sinusoidal parameters are (duration=0.5s) and (amplitude=0.48m)

C. Balance

One intrinsic advantage of balancing the pendulum with 2R arm is that, it enables 2 degree of freedoms, horizontal and vertical, for the balance motion. Yet, this could involve a very complex system dynamics which makes it hard to model. Therefore, to facilitate the problem, the endeffector of the 2R arm is only allowed to move in horizontal direction. Then, this balance motion can be approximated into a classic cart-pole balance problem, and a state-space controller is selected to balance the pendulum. For the cart-pole system, the governing equations, and the state-space form of the transfer functions are shown below:

$$\left(I + ml^2\right)\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{6}$$

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \tag{7}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \tag{8}$$

$$\ddot{X} = A\dot{X} + Bu \tag{9}$$

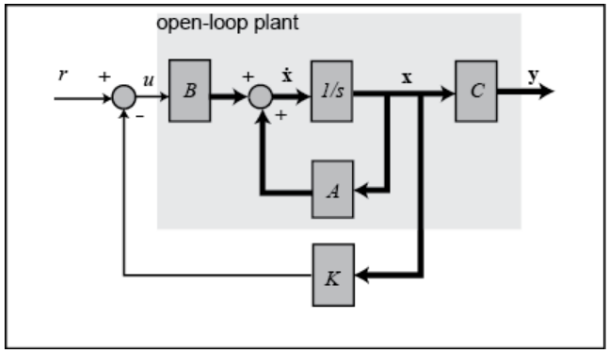
$$\ddot{X} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} \tag{10}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-\left(I+ml^2\right)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} \tag{12}$$

where **m** is the mass of the pendulum, **M** is the mass of the cart, **I** is the moment of inertia of the pendulum, **l** is the link length, ϕ is the disturbance angle from the inverted position, **x** is the displacement of the cart.

However, finding a value of **M** to precisely represent the overall mass of the endeffector with respect to the pendulum joint is a rather difficult task, since we can only derive a 2×2 mass matrix of the endeffector rather than a desired scalar value. Therefore, a physical assumption is made that the 2R arm is completely decoupled with the pendulum. With this assumption, **M** is set to be 0, and the external force will only affects the dynamics of the pendulum. A block diagram of implementing Eq.9 and Eq.8 is shown below.



[3]

Fig. 6: Block diagram of a State Space Controller

Since the state-space controller is designed to integrate with the inverse dynamic controller, the output of the state-space controller need to be carefully selected to become the input of the inverse dynamics controller. From the block diagram, the input **r** represents the reference position vector including 4 states, x, $\dot{x}$, ϕ, $\dot{\phi}$,and the output **y** includes 2 terms, displacement x and joint angle disturbance ϕ. Since joint angle is really not a valid input of the inverse dynamics, displacement x is the only needed result from the state-space controller. In this case, the output of the state-space controller would be used as the **desired position input** of the inverse dynamics.

Another possible way is to use the intermediate result $\dot{X}$ as the input of the inverse dynamics. The original $\dot{X}$ consists of four terms $\dot{x}$, $\ddot{x}$, $\dot{\phi}$, $\ddot{\phi}$. While $\dot{x}$ is just the initial value as shown from Eq.11 and Eq.9, $\dot{\phi}$ and $\ddot{\phi}$ is not a valid input of the inverse dynamics controller. Therefore, $\dot{X}$ will only includes the transnational acceleration term, $\ddot{x}$. In this case, the output of the state-space controller would be used as the **desired acceleration input** of the inverse dynamics.

For this project, the second method is chosen, since the first one requires another time integration from $\dot{X}$ to X, which results to totally two time-integration for the overall simulation. The simplified, feed-forward block diagram of the balancing system using the second method is shown below.

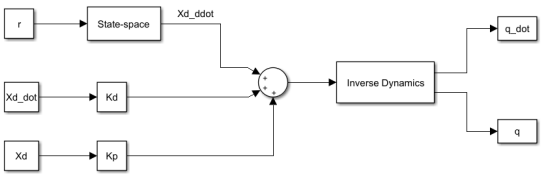


Fig. 7: Simplified block diagram of the overall system

Notice the state-space controller does not activate until the manipulator reaches at a certain configuration that can be balanced during the swing-up motion. The only system requirement for the state-space controller is to reach the desired position and stay stationary with the pendulum inverted.

IV. Results

A. Inverse Dynamics

To test the response of the inverse dynamics controller a step input was applied to both the X and Y position for the manipulator. The manipulator is initially at coordinates(0.0,0.75) and is commanded to (0.5,1.25).

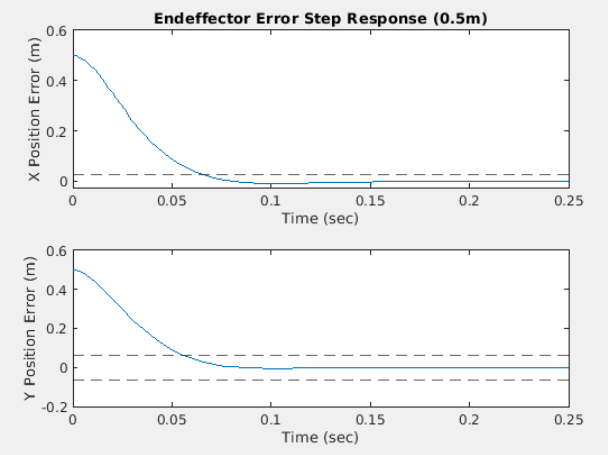


Fig. 8: Plot showing the step response of the manipulator $\delta X = \delta Y = 0.5$. Dotted lines indicate bounds of 5% error

The step response analysis shows that the settling time for X and Y of 0.0735s and 0.0752s respectively. This clearly meets the required performance specifications determined earlier. The gains chosen for this controller are based on a 2nd order response of natural frequency $\omega_n = 50rad/s$ and a dampening ratio of $\zeta = 0.8$. Then assuming that the controller resulted in a decoupled system stabilized by the proportional and derivative control action, initial gains can be found using $diag(K_p) = \omega_n^2$ and $diag(K_d) = 2\zeta\omega_n$ [2].

B. Swing Up

The swing up controller shown below in Fig.9 demonstrates the ability for the inverse dynamics controller to track the reference trajectory well enough for the pendulum to achieve a upright orientation within the required time of 1 second. From Fig.9 it can be seen that the pendulum measured by ϕ enters the balance controller domain within 0.9seconds. The balance controller domain is shown by the dotted lines in Fig.9 representing bounds of $-0.3 < \phi < 0.3$. From the time 0-0.5s marks the forced trajectory portion of swing up, the following 0.5 - 1.0s marks the uncontrolled swing portion where the manipulator attempts to remain stationary while the pendulum gently enters the balance domain.

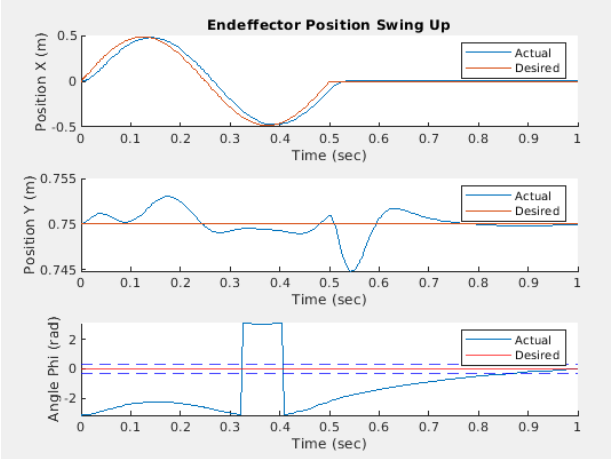


Fig. 9: Error plots during the swing up routine, sinusoidal trajectory duration of (0.5s) and an amplitude of (0.485m)

C. Balance

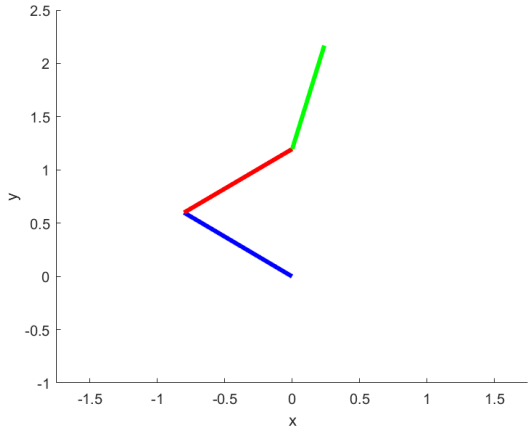


Fig. 10: Initial configuration for balance simulation

The figure above is the initial configuration selected for the balance simulation. This configuration is intentionally selected with slow joint velocity and small disturbance for quick balance. The gain matrix K of the state-space controller is initially found using Linear Quadratic Regulation, setting the displacement and joint angle with equal weight, 100. The result of 15s balance simulation is shown below.

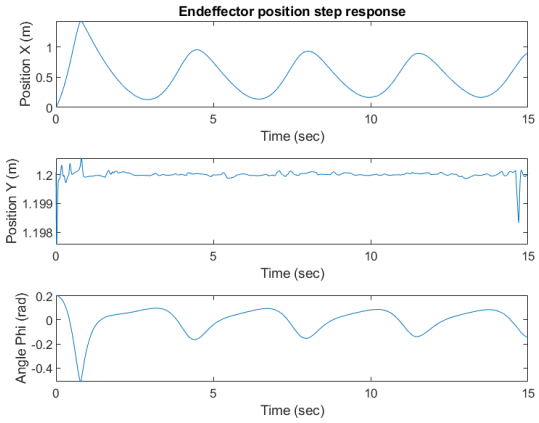


Fig. 11: 15s balance simulation with untuned gain K

From Fig.11, without tuning the gain matrix K, even though the state-space controller does work, the end-effector position keeps oscillating without showing any obvious convergence. Therefore, the gain matrix K is manually tuned by enlarging the second entry of K for about 1.5 times. Its corresponding simulation result is shown below.

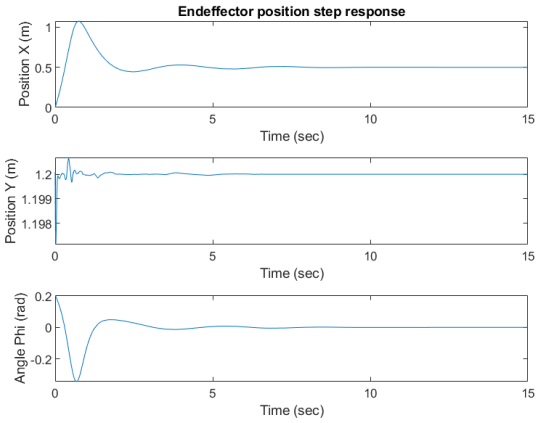


Fig. 12: 15s balance simulation with tuned gain K

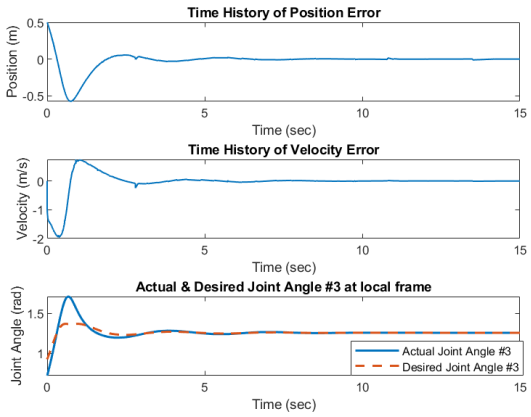


Fig. 13: Error Assessment of tuned gain K

As the second entry of K is enlarged, the amplitude of oscillations decrease significantly, and the system is able to achieve the steady state at about 8th second. Second term of the gain matrix K corresponds to the derivative term of the pendulum position. Fig.13 shows that all the error terms converge to zero at the steady state, which satisfies the system requirement.

For this specific swing-up and balance simulation, the pendulum did not start to be balanced until multiple 360^o rotations after the swing-up. It is because, for this particular simulation, the cubic polynomial swing-up controller was used, with a relative high velocity during the movement. The sudden stop,when the endeffector

reached to desired position, gave the pendulum large amount of kinetic energy which causing the pendulum to rotate with a high joint velocity. The joint velocity of the pendulum was gradually decreased during the rotation due to the joint friction. This simulation is used for testing the capability of the balance controller. In a formal simulation, when a properly tuned swing up controller such as sinusoidal trajectory was used, multiple rotations would not be necessary at all.

V. Discussion

A. Inverse Dynamics

The inverse dynamics controller plays a critical role in ensuring that the dynamics of the manipulator are sufficiently compensated for. For this specific task the dynamics of the pendulum also influence the dynamics of the manipulator. To better understand the influence of the pendulum a controller which only partially compensates the the dynamics of the 3 link system was implemented. This partial compensation used a separate model to describe only the manipulator within the 3 link system. The same gains used in the fully compensated controller were used for the partially compensated controller to compare performance. As seen in Fig.14 the partially compensated controller fails to meet the performance criteria as it crosses the error bounds and takes too long to settle. This is not surprising as the gains are not tuned for this controller. Regardless of tuning effort on a controller such as this, the non-linear dynamics of the pendulum link will always be seen as a disturbance to this partially compensated manipulator see Fig.15. It should also be noted that as the gains are increased the on the partially compensated controller the errors are also increased. This is to be expected as the un-modelled dynamics of the pendulum will have a larger influence while the manipulator is operating at higher velocities and accelerations. For that reason the full 3-link compensation was selected for the controller. A direct comparison between the two partial and full compensation controllers can be seen from both Fig.15 & Fig.16 respectively.

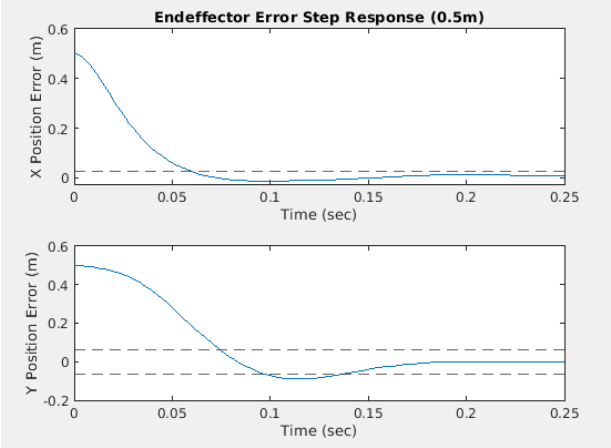


Fig. 14: Step response(0.5m) of manipulator with inverse dynamics controller only compensating for manipulator dynamics

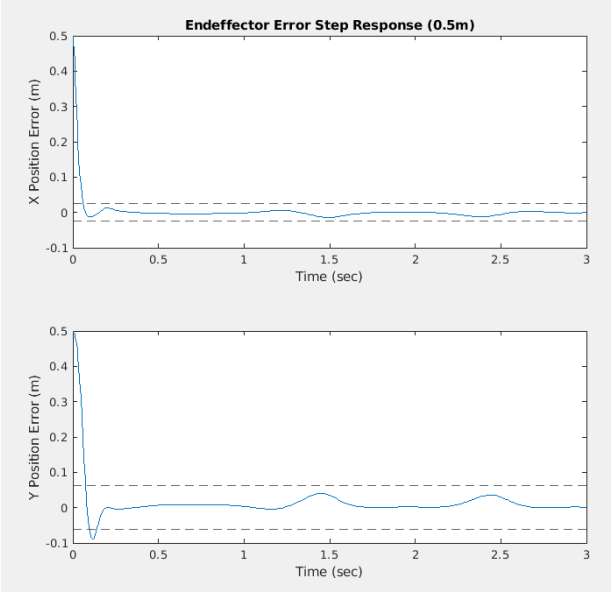


Fig. 15: Step response(0.5m) across 3s showing the disturbance of the pendulum dynamics on the EE position over time

B. Swing Up

The swing up controller design has to consider more than just moving the pendulum to the upright position. This became even more obvious while integrating the two controllers together such that the pendulum can be stabilized. The swing up controller is ultimately responsible for ensuring the pendulum state is within the stability conditions of the balance controller. Initially the swing up controller was designed such that depending on the length of the pendulum a trajectory could be generated to deliver the pendulum to the vertical position with nearly zero angular velocity. This controller was essentially a trajectory generator whose parameters relied on a the final velocity determined by the following

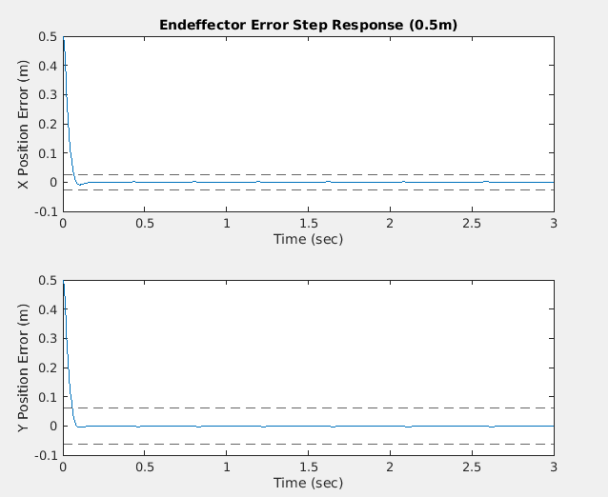


Fig. 16: Step response(0.5m) across 3s showing the compensation of the pendulum dynamics

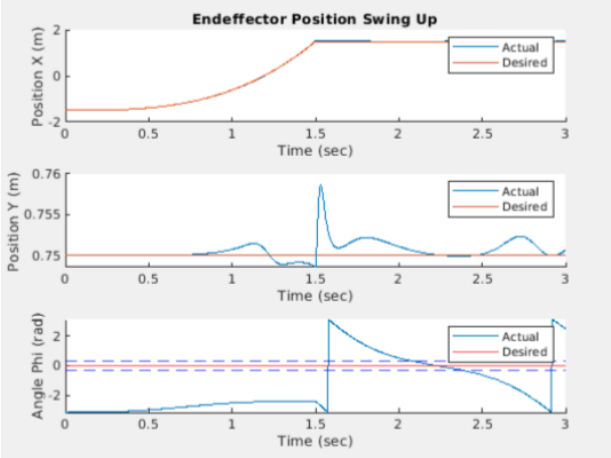


Fig. 17: Error plot showing the swing up trajectory which accelerates the pendulum in one direction

equation $V_f = \sqrt{2gl}$ which can be derived from an energy analysis of the pendulum assuming no work done by non-conservative forces between the time the swing up trajectory ended and the pendulum reaches the apex of its swing. This open-loop controller had several drawbacks. It was assumed that by the end of the trajectory the pendulum would be hanging down vertically. The final velocity of the trajectory was non-zero and had to start from rest, the acceleration that the pendulum would experience resulted in the pendulum being dragged through the trajectory so that when the manipulator stopped the pendulum was horizontal and all the kinetic energy in the pendulum would be transferred in to the manipulator. Additionally this style of swing up resulted in the pendulum being swung up near the edge of the workspace which increased the Chance of the balance controller failing from encountering a singularity. Due to the complexity of accounting for all these influences it was challenging to tune the controller as can be seen in Fig.17 where the 3rd plot shows a high pendulum velocity entering the the error bounds. Ultimately this trajectory was replaced by a more compact trajectory generator which uses sinusoids to drive the end of the manipulator as was discussed earlier.

C. Balance

1) Challenges: A few challenges are faced during the balance simulation. The first challenge is limited horizontal mobility of the endeffector. Just like the cart which can only move within a specified range, the range of movement of the endeffector is limited by its link length. If the endeffector stretch too far in order to balance the pendulum, the manipulator will eventually reach at the singularity condition, where the links of 2R arm are aligned, and fails to balance. Therefore, the initial joint velocity and the disturbance angle can not be too large, and the desired final position can not be too far away from the origin, so that the manipulator can balance the pendulum within its workspace.

Another challenge is the decoupling compensation. Notice for the state-space controller, the pendulum is assumed to be perfectly decoupled with the manipulator in order to facilitate computation. However, the output of the **decentralized** state-space controller is used as the input of the **centralized** inverse dynamic controller as shown in Fig7. This inconsistent physical model between two controllers make the balance motion of the pendulum hard to predict. For instance, in an ideal condition which the state-space controller and the inverse dynamic controller are both well-centralized, then the gain matrix K from the LQR should be the optimal gain matrix which balances the pendulum and reaches steady state quickly. However, since the state-space controller is actually decentralized, the gain matrix K from LQR does not give its corresponding system behaviour, just like the example shown in Fig11. Therefore, in order to compensate for the decoupling, the gain matrix K need to be manually tuned.

The third challenge is to verify the stability of the system. This challenge can be regarded as a collaborative result of previous two challenges. Assuming for an arbi-

trary initial configuration, the balance simulation failed because the manipulator reached its singularity. It was hard to tell what actually caused that failure. High initial velocity, disturbance, and risky desired final position can all possibly cause the failure even though the system is stable. Or, the system is simply unstable due to the non-optimal gain matrix K from LQR which cause the failure no matter what initial velocity, disturbance and final position are chosen. Therefore, the system is regarded as **"vulnerable"** due to the multiple factors that can lead to balance failure.

2) Solutions: In order to address these challenges and verify if the system itself is stable, the following solution steps are structured.

1) Simulate the swing-up motion. Find if there exists at least one configuration with low initial velocity and small disturbance during the whole swing-up motion.
2) If no such configuration exists, adjust the swing-up motion and return to step 1.
3) If such configuration exists, extract its data including joint angles, joint velocities. Use the data as the initial condition for the balance simulation.
4) If balance fails, refine the requirement of initial velocity and disturbance, and return to step 1.
5) If balance simulation proceed successfully, run the integrated simulation, and tune the gain matrix K for better performance.

Following such steps can minimize the influence of initial velocity and disturbance on balance failure. In addition, another strong advantage is that, the satisfied configuration found for balancing can always be reproduced later during the integrated simulation. For this project, as the total joint velocity of 3 links are smaller than 2.5 rad/s, and the disturbance is smaller than 0.4 radian, the balance system can always achieve at least marginal stability.

3) Gain Matrix K: In order to improve the system performance, gain matrix K need to tuned. The gain matrix K from LQR has a 1×4 dimension. From Fig.6, K is multiplied with **X** in the feedback loop. To find the influence of each entry of K on the system behaviour, the equation derivation is shown below.

$$u = -K\dot{X} \tag{13}$$

$$\ddot{X} = (A - BK)\dot{X} \tag{14}$$

$$K = \begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix} \tag{15}$$

Rewrite the Eq.12 as

$$B = \begin{bmatrix} 0 \\ b_1 \\ 0 \\ b_2 \end{bmatrix} \tag{16}$$

$$BK = \begin{bmatrix} 0 & 0 & 0 & 0 \\ b_1k_1 & b_1k_2 & b_1k_3 & b_1k_4 \\ 0 & 0 & 0 & 0 \\ b_2k_1 & b_2k_2 & b_2k_3 & b_2k_4 \end{bmatrix} \tag{17}$$

Since only the relationship between output $\ddot{X}$ and gain matrix K is interested, assume A = 0 for the Eq.14, and that equation can be rewrite as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ b_1k_1 & b_1k_2 & b_1k_3 & b_1k_4 \\ 0 & 0 & 0 & 0 \\ b_2k_1 & b_2k_2 & b_2k_3 & b_2k_4 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ b_1(k_1x + k_2\dot{x} + k_3\phi + k_4\dot{\phi}) \\ 0 \\ b_2(k_1x + k_2\dot{x} + k_3\phi + k_4\dot{\phi}) \end{bmatrix} \tag{19}$$

From the Eq.19, the translational acceleration is actually affected by the summation of the product of each state and its corresponding gain k_i. Therefore, when tuning the gain matrix, the sign of each gain should keep as the same, and only change the magnitude.

VI. Conclusion

Throughout the development of this project many challenges were overcome and valuable learning's attained. Some of the challenges were a result of the underlying mechanics behind solving such a problem; having less to do with understanding class material and more about gaining experience in simulating complex control system. Simulating the full system that discretely switched between controllers is computationally dubious. Depending on the solver the simulation may be impossible to solve. One solution was to use a non-stiff solver such as "ode23tb()". This solver adapts well to a dynamic system whose model parameters are not smooth over time, but rather can rapidly change in orders of magnitude as was the case when the full-state feedback

controller is engaged. Additionally tuning of the full-state feedback controller took a considerable amount of time as each simulation took minutes to run and the influence of each gain on the stability could only be explored manually as our decoupled assumption resulted in slight inconsistencies between stable gains and the results of the LQR method. Using the LQR gain values as a starting point and understanding how to stabilize the system was a satisfying accomplishment. Enabling the full-state feedback controller to follow trajectories is certainly a subject to investigate further. Based on the work presented here it has been demonstrated that a full-state feedback controller can stabilize a system under the assumption of decoupled dynamics in a 3R system with a 2R manipulator.

VII. Future Works

As work on the project extends into the future there are certain questions that should be answered. How can a full-state feedback controller be designed to stabilize the pendulum while tracking a trajectory?. This work might involve the adoption of a more advanced controller scheme such as model predictive control. The dynamics used in this project were developed so that a sense of realism and complexity were captured, this lends itself to not only simulation but also implementing this control system on a real system. Developing a real-time version of these controllers would enable further learning's regarding computational speed and accuracy. Additionally redeveloping the swing up controller to be closed loop utilizing a nonlinear control scheme is highly desirable.

References

[1] Fuat Peker et al. "Cascade Control Approach for a Cart Inverted Pendulum System Using Controller Synthesis Method". In: *2018 26th Mediterranean Conference on Control and Automation (MED)*. 2018 26th Mediterranean Conference on Control and Automation (MED). ISSN: 2473-3504. June 2018, pp. 1–6. DOI: 10.1109/MED.2018.8442552.

[2] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. Jan. 1, 2011. ISBN: 978-1-84628-641-4. DOI: 10.1007/978-1-84628-642-1.

[3] Dawn Tilbury. *Inverted Pendulum: System Modeling*. 2011. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling.

VIII. Appendix

A. Work distribution

Jonathan Bopp

1) Derived equations of motion
2) Developed swing up controller
3) Developed inverse dynamics controller

Rong Jin

1) Developed full-state feedback controller
2) Tuned full-state controller based on LQR
3) Set logic to switch controllers in integrated simulation.

B. Code

```
function zdot = manipulator(t, z_0,B_f,N_f)
theta_1 = z_0(1);
theta_2 = z_0(2);
theta_3 = z_0(3);
theta_dot_1 = z_0(4);
theta_dot_2 = z_0(5);
theta_dot_3 = z_0(6);

params;

% Inertia matrix
%mat = load('Mass_matrix.mat');
var = [I_l1, I_l2, I_l3, I_m1, I_m2, a_1, a_2, kr_1, kr_2, l_1, l_2, l_3, m_l1, m_l2, m
B = subs(B_f,symvar(B_f),var);
%B = diag(diag(B));
% move sym functions

% Nonlinear term
%mat = load('N_term.mat');
var = [a_1, a_2, g, l_1, l_2, l_3, m_l1, m_l2, m_l3, m_m2, theta_1, theta_2, theta_3, t
N = subs(N_f,symvar(N_f),var);

B_f = double(vpa(B,3));
N_f = double(vpa(N,3));

% Inertia matrix
b11 = I_l1 + m_l1*l_1^2+kr_1^2*I_m1+I_l2+m_l2*(a_1^2+l_2^2+2*a_1*l_2*cos(theta_2)) + I_
b12 = I_l2+m_l2*(l_2^2+a_1*l_2*cos(theta_2))+kr_2*I_m2;
b21 = b12;
b22 = I_l2+ m_l2*l_2^2+kr_2^2*I_m2;

% Christoffel symbols
c111 = 0;
c112 = -m_l2*a_1*l_2*sin(theta_2);
c122 = c112;
c211 = -c112;
c212 = 0;
c222 = 0;
h = c112;

% Gravity terms
g11 = (m_l1*l_1+m_m2*a_1+m_l2*a_1)*g*cos(theta_1) + m_l2*l_2*g*cos(theta_1+theta_2);
g22 = m_l2*l_2*g*cos(theta_1+theta_2);

% Assemble manipulator dynamics
B_a = [b11 b12; b21 b22];
C_a = [h*theta_dot_2 h*(theta_dot_1+theta_dot_2); -h*theta_dot_1 0];
%N_a = [2*h*theta_dot_2 h*theta_dot_2; h*theta_dot_2 0]-2*[h*theta_dot_2 h*(theta_dot_1
G_a = [g11; g22];

% Jacobian Analytical
j11 = -a_2*sin(theta_1+theta_2)-a_1*sin(theta_1);
j12 = -a_2*sin(theta_1+theta_2);
```

```
j21 = a_2*cos(theta_1+theta_2)+a_1*cos(theta_1);
j22 = a_2*cos(theta_1+theta_2);
Ja = [j11 j12; j21 j22];

% Derivative of Jacobian Analytical
s1 = a_2*sin(theta_1+theta_2)*(theta_dot_1+theta_dot_2);
s2 = a_2*cos(theta_1+theta_2)*(theta_dot_1+theta_dot_2);
jd11 = -s2-a_1*cos(theta_1)*theta_dot_1;
jd12 = -s2;
jd21 = -s1-a_1*sin(theta_1)*theta_dot_1;
jd22 = -s1;
Ja_dot = [jd11 jd12; jd21 jd22];

% Define joint variable
q = [theta_1 theta_2 theta_3]';
q_arm = [theta_1 theta_2]';
q_dot = [theta_dot_1 theta_dot_2 theta_dot_3]';
q_dot_arm = [theta_dot_1 theta_dot_2]';

phi = wrapToPi((theta_1 + theta_2 + theta_3) - pi/2); % Pendulum measured from pos y
phi_dot = theta_dot_3;

% Get desired variables
Xd = [0; 0.75];
Xd_dot = [0;0];
Xd_ddot = [0;0];

% Trajectory from 4 to 7
if t > 4.0
    Xi = [0.0; 0.75];
    Xf = Xi + [0.250; 0.250];
    Vi = [0;0];
    Vf = [0;0];
    [x v a] = ContinousCPTraj(Xi,Xf,Vi,Vf,4,3,t);

    Xd = x;
    Xd_dot = v;
    Xd_ddot = a;
end



%if t > 3
%    Xd =
%    Xd_dot =
%    Xd_ddot =
%end


% Define operational space variables
Xe = fk(q_arm);
Xe_dot = Ja*q_dot_arm;

if abs(phi) < 0.3 | t > 0.85
```

```
    SWING_UP = false;
else
    SWING_UP = true;
end

%SWING_UP = true;

if SWING_UP == true
    % Get trajectory values
%     Xi = [-1.5; 0.75];
%     Xf = Xi + [3.0; 0.0];
%     Vi = [0;0];
%     Vf = [6.0;0];
%     [x,v,a] = ContinousCPTraj(Xi,Xf,Vi,Vf,1.5,t);
    [x,v,a] = SinusoidalSwingUp(0.485,0.5,t);
    Xd(1) = x(1);
    Xd_dot(1) = v(1);
    Xd_ddot(1) = a(1);
end

if SWING_UP == false
    % Pendulum control Full State Feedback
    K_pend = [-100 -200 500 100];
    R_pend = [Xd(1) Xd_dot(1) 0 0]';
    X_pend = [Xe(1) Xe_dot(1) phi phi_dot]';
    U_pend = Xd(1) - K_pend*(X_pend);
    %Xd_dot(1) = U_pend;
    Xd_ddot(1) = U_pend;
    Kd(1,1) = Kd(1,1)*0; % kill manipulator velocity control
    Kp(1,1) = Kp(1,1)*0; % kill manipulator position control
end

% Calculate manipulator error
X_tilda = Xd-Xe;
X_tilda_dot = Xd_dot-Xe_dot;

% Inverse dynamics operational space controller
y_arm = Ja\(Xd_ddot+Kd*X_tilda_dot+Kp*X_tilda-Ja_dot*q_dot_arm);
y = [y_arm(1) y_arm(2) 0]';
% Inverse dynamics linearizing control
% Controller compensating only arm
%u = B_a*y_arm + C_a*q_dot_arm + G_a;
% Controller compensating full
u = B_f*y + N_f;

%u = 0;
u = [u(1) u(2) 0]';
if t > 10
    u = [0 0 0]';
end

% Full Model
RHS =-N_f -Fv*q_dot + u;
LHS = B_f;
```

```
q_ddot = double(LHS\RHS);
theta_ddot_1 = q_ddot(1);
theta_ddot_2 = q_ddot(2);
theta_ddot_3 = q_ddot(3);

Xd
Xd_dot
Xd_ddot
phi
theta_dot_3
t
SWING_UP

zdot = [theta_dot_1; theta_dot_2; theta_dot_3; theta_ddot_1; theta_ddot_2; theta_ddot_3
end
```

```
close all
clear all
clc
```

The goal of this script is to derive the dynamic equations for a 3-link manipulator with only the two first links being actuated, this means the system has 3 links with mass and inertia, but only 2 motors with mass, inertia, and gear reductions

```
% Link masses
m_l1 = 1.0;
m_l2 = 1.0;
m_l3 = 1.0;

% Link length
a_1 = 1;
a_2 = 1;
a_3 = 1;

% Motor masses
m_m1 = 0.25;
m_m2 = 0.25;

% Gear reductions
kr_1 = 1;
kr_2 = 1;

% Distance to center of mass
l_1 = a_1/2;
l_2 = a_2/2;
l_3 = a_3/2;

% Link inertia
I_l1 = (1/12)*m_l1*l_1^2;
I_l2 = (1/12)*m_l2*l_2^2;
I_l3 = (1/12)*m_l3*l_3^2;

% Motor ineria
I_m1 = 0.1;
I_m2 = 0.1;

% Enviornment parameters
%g = 9.81; % m/s^2
```

```
syms g real
syms theta_1 real
syms theta_2 real
syms theta_3 real
```

```
syms theta_dot_1 real
syms theta_dot_2 real
syms theta_dot_3 real
syms m_l1 m_l2 m_l3 m_m1 m_m2 real
syms l_1 l_2 l_3 I_l1 I_l2 I_l3 I_m1 I_m2 real
syms a_1 a_2 a_3 real
syms kr_1 kr_2 real
syms g real

%syms m_l1 m_l2 m_l3 m_m1 m_m2 l_1 l_2 l_3 I_l1 I_l2 I_l3 I_m1 I_m2 a_1 a_2 a_3 kr_1 kr

% theta_dot_1 = diff(theta_1,t);
% theta_dot_2 = diff(theta_2,t);
% theta_dot_3 = diff(theta_3,t);

%assume(theta_dot_1,'real');
%assume(theta_dot_2,'real');
%assume(theta_dot_3,'real');

T0_1 = dhTransform(0,0,theta_1,0);
T1_2 = dhTransform(0,a_1,theta_2,0);
T2_3 = dhTransform(0,a_2,theta_3,0);
T3_E = dhTransform(0,a_3,0,0);

T0_E = formula(T0_1*T1_2*T2_3*T3_E);
T0_3 = formula(T0_1*T1_2*T2_3);
T0_2 = formula(T0_1*T1_2);
T0_1
T0_1 = T0_1
R0_1 = T0_1(1:3,1:3);
R0_2 = T0_2(1:3,1:3);
R0_3 = T0_3(1:3,1:3);

% Define Positions along arm
p_l1_s = T0_1*[l_1 0 0 1]'
p_l2_s = T0_2*[l_2 0 0 1]'
p_l3_s = T0_3*[l_3 0 0 1]'

p_m1_s = T0_1*[0 0 0 1]';
p_m2_s = T0_2*[0 0 0 1]';

pl1 = p_l1_s(1:3);
pl2 = p_l2_s(1:3);
pl3 = p_l3_s(1:3);

pm1 = p_m1_s(1:3);
pm2 = p_m2_s(1:3);

% Define Positions to Joints
p0 = [0 0 0]';
p1 = T0_2(1:3,4);
p2 = T0_3(1:3,4);
p3 = T0_E(1:3,4);
```

```
Z0  = [0 0 1]';
Zeros = [0 0 0]';

Jl1_P = [cross(Z0,pl1-p0), Zeros, Zeros];
Jl1_O = [Z0, Zeros, Zeros];
Jl1 = [Jl1_P; Jl1_O]

Jl2_P = [cross(Z0,pl2-p0),cross(Z0,pl2-p1),Zeros];
Jl2_O = [Z0, Z0, Zeros];
Jl2 = [Jl2_P; Jl2_O]

Jl3_P = [cross(Z0,pl3-p0),cross(Z0,pl3-p1),cross(Z0,pl3-p2)];
Jl3_O = [Z0,Z0,Z0];
Jl3 = [Jl3_P; Jl3_O]

Jm1_P = [Zeros, Zeros, Zeros];
Jm1_O = [Z0*kr_1, Zeros, Zeros];
Jm1 = [Jm1_P; Jm1_O]

Jm2_P = [cross(Z0,pm2-p0),Zeros, Zeros];
Jm2_O = [Z0, Z0*kr_2, Zeros];
Jm2 = [Jm2_P; Jm2_O]

q_dot = [theta_dot_1 theta_dot_2 theta_dot_3]';

% Velocities of link center of masses
P_dot_l1 = Jl1_P*q_dot;
P_dot_l2 = Jl2_P*q_dot;
P_dot_l3 = Jl3_P*q_dot;

% Angular Velocities of links
O_dot_l1 = Jl1_O*q_dot;
O_dot_l2 = Jl2_O*q_dot;
O_dot_l3 = Jl3_O*q_dot;

% Velocities of the motors
P_dot_m1 = Jm1_P*q_dot;
P_dot_m2 = Jm2_P*q_dot;

% Angular velocities of the motors
O_dot_m1 = kr_1*theta_dot_1*Z0;
O_dot_m2 = kr_2*theta_dot_2*Z0;

% Kinetic Energy of the links
T_li(1) = 1/2*m_l1*(P_dot_l1')*P_dot_l1+1/2*(O_dot_l1')*R0_1*I_l1*R0_1'*O_dot_l1;
T_li(2) = 1/2*m_l2*(P_dot_l2')*P_dot_l2+1/2*(O_dot_l2')*R0_2*I_l2*R0_2'*O_dot_l2;
T_li(3) = 1/2*m_l3*(P_dot_l3')*P_dot_l3+1/2*(O_dot_l3')*R0_3*I_l3*R0_3'*O_dot_l3;

% Kinetic Energy of the motors
T_mi(1) = 1/2*m_m1*(P_dot_m1')*P_dot_m1+1/2*(O_dot_m1')*I_m1*O_dot_m1;
T_mi(2) = 1/2*m_m2*(P_dot_m2')*P_dot_m2+1/2*(O_dot_m2')*I_m2*O_dot_m2;

Ti(1) = T_li(1)+T_mi(1);
Ti(2) = T_li(2)+T_mi(2);
```

```
Ti(3) = T_li(3);
simplify(Ti');

G = [0 -g 0]';
% Potential Energy of links
Ui(1) = -(m_m1*G'*pm1 + m_l1*G'*pl1);
Ui(2) = -(m_m2*G'*pm2 + m_l2*G'*pl2);
Ui(3) = -(m_l3*G'*pl3);
simplify(Ui');
U = Ui(1) + Ui(2) + Ui(3);
T = Ti(1) + Ti(2) + Ti(3);
```

```
X = {theta_1 theta_dot_1 theta_2 theta_dot_2 theta_3 theta_dot_3}
L=T-U;
syms tau1 tau2 tau3 real
Qe = {tau1 tau2 tau3};
Qi = {0 0 0};
par = {m_l1 m_l2 m_l3 m_m1 m_m2 l_1 l_2 l_3 I_l1 I_l2 I_l3 I_m1 I_m2 a_1 a_2 a_3 kr_1 k
Eqn = EulerLagrange(L,X,Qi,Qe,0,par,'m')
```

```
syms theta_ddot_1 theta_ddot_2 theta_ddot_3 real
tau = [tau1; tau2; tau3];
Eq =[Eqn(2) == theta_ddot_1
     Eqn(4) == theta_ddot_2
     Eqn(6) == theta_ddot_3];
```

```
%[,A] = collect(Eq,{theta_ddot_1 theta_ddot_2 theta_ddot_3})
q_ddot = [theta_ddot_1 theta_ddot_2 theta_ddot_3]';
[W,E] = equationsToMatrix(Eq,[tau1 tau2 tau3])
lhs = simplify(W\E);
rhs = [tau1 tau2 tau3]';

[Mass_matrix,Other] = equationsToMatrix(lhs,[theta_ddot_1 theta_ddot_2 theta_ddot_3])
Mass_matrix
%[C_matrix,Else] = equationsToMatrix(Other,[theta_dot_1 theta_dot_2 theta_dot_3])
N_matrix = simplify(-Other)
N_iso = [N_matrix(1) == 0
         N_matrix(2) == 0
         N_matrix(3) == 0];
%out = equationsToMatrix(N_iso,[theta_dot_1 theta_dot_2 theta_dot_3])

EOM_rough = lhs;
EOM_clean = Mass_matrix*q_ddot + N_matrix ;
isequal(EOM_rough(1),EOM_clean(1))
simplify(EOM_rough,"Steps",100)
simplify(EOM_clean,"Steps",100)
save('Mass_matrix.mat','Mass_matrix')
save('N_term.mat','N_matrix')
```

```
 % Assemble B Matrix
% Shortcut derivation of the mass matrix
```

```
B1 = m_l1*(Jl1_P')*Jl1_P+Jl1_O'*R0_1*I_l1*R0_1'*Jl1_O + m_m1*(Jm1_P')*Jm1_P+Jm1_O'*R0_1
B2 = m_l2*(Jl2_P')*Jl2_P+Jl2_O'*R0_2*I_l2*R0_2'*Jl2_O + m_m2*(Jm2_P')*Jm2_P+Jm2_O'*R0_2
B3 = m_l3*(Jl3_P')*Jl3_P+Jl3_O'*R0_3*I_l3*R0_3'*Jl3_O;
B = B1 + B2 + B3;
```

```
close all
clear all
clc

tic
params;

% Initial conditions
disturbance = -0.0;
qi = ik(0,0.75)
theta_1_0 = qi(1);
theta_2_0 = qi(2);
theta_3_0 = wrapToPi(-(qi(1)+qi(2)+pi/2+disturbance));
theta_dot_1_0 = 0;
theta_dot_2_0 = 0;
theta_dot_3_0 = 0;

tspan = [0 15];

% ODE45 settings
options = odeset('RelTol',1e-3,'AbsTol',1e-5);
%options = odeset('reltol', 1E-7);
z_0 = [theta_1_0; theta_2_0; theta_3_0; theta_dot_1_0; theta_dot_2_0; theta_dot_3_0];

B_mat = load('Mass_matrix.mat');
B_f = B_mat.Mass_matrix;
N_mat = load('N_term.mat');
N_f = N_mat.N_matrix;

[time, z] = ode23tb(@(t,y) manipulator(t,y,B_f,N_f), tspan, z_0, options);

toc
```

Get varibles from simulation

```
theta_1 = z(:,1);
theta_2 = z(:,2);
theta_3 = z(:,3);

q = [theta_1 theta_2 theta_3]';
phi = wrapToPi(-q(1,:)-q(2,:)-q(3,:)+pi/2);
for i = 1:length(time)
    Xe(:,i) = fk(q(1:2,i));
end

hold on
plot(time,theta_1)
plot(time,theta_2)
plot(time,theta_3)
hold off

figure
```

```
subplot(3,1,1)
plot(time,Xe(1,:))
title("Endeffector position step response (Balance Controller)")
ylabel("Position X (m)")
xlabel("Time (sec)")

subplot(3,1,2)
plot(time,Xe(2,:))
ylabel("Position Y (m)")
xlabel("Time (sec)")
%axis([0 tspan(1,2) 0.71 0.79])

subplot(3,1,3)
plot(time,phi)
ylabel("Angle Phi (rad)")
xlabel("Time (sec)")

% Calculating link endpoints
L1_x = a_1*cos(theta_1);
L1_y = a_1*sin(theta_1);
L2_x = L1_x + a_2*cos(theta_1+theta_2);
L2_y = L1_y + a_2*sin(theta_1+theta_2);
L3_x = L2_x + a_3*cos(theta_1+theta_2+theta_3);
L3_y = L2_y + a_3*sin(theta_1+theta_2+theta_3);

[L1_x ts] = resample(L1_x,time);
[L1_y ts] = resample(L1_y,time);
[L2_x ts] = resample(L2_x,time);
[L2_y ts] = resample(L2_y,time);
[L3_x ts] = resample(L3_x,time);
[L3_y ts] = resample(L3_y,time);

figure()
axis([-1.75 1.75 -1 2.5]);
xlabel('x');
ylabel('y');

L1_line = line('xdata', [0, L1_x(1)], 'ydata', [0, L1_y(1)],...
    'linewidth', 3, 'color', 'blue');
L2_line = line('xdata', [L1_x(1), L2_x(1)], 'ydata', [L1_y(1), L2_y(1)],...
    'linewidth', 3, 'color', 'red');
L3_line = line('xdata', [L2_x(1), L3_x(1)], 'ydata', [L2_y(1), L3_y(1)],...
    'linewidth', 3, 'color', 'green');
```

```
delay = tspan(1,2)/length(ts);
hold on
%yline(0.75);
%xline(0);
grid on
v = VideoWriter('Output.avi');
open(v);
% Draw and redraw the objects with new xdata, ydata
for i=1:length(ts)
```

```
    pause(delay)
    set(L1_line, 'xdata', [      0, L1_x(i)], 'ydata', [      0, L1_y(i)]);
    set(L2_line, 'xdata', [L1_x(i), L2_x(i)], 'ydata', [L1_y(i), L2_y(i)]);
    set(L3_line, 'xdata', [L2_x(i), L3_x(i)], 'ydata', [L2_y(i), L3_y(i)]);
    frame = getframe(gcf);
    writeVideo(v,frame);
    drawnow;
end

hold off
close(v)
```