Nonlinear Systems Final Project Feedback (2023 Spring) & F.A.Q.

February 18, 2025

1 General Advice

- **General Advice for Hardware Testing:**
  - Be prepared for discrepancies between simulation and hardware. Factors like static friction, sensor noise, and un-modeled dynamics can cause significant differences in performance.
  - Prioritize early testing on hardware to identify bugs early and adjust your models or controllers accordingly.
  - Re-tuning of controller parameters may be necessary when transitioning from simulation to hardware.
  - Hardware testing can be a challenging but valuable learning experience. Expect that the process may require reworking controllers and estimators.
- **Controller Selection and Tuning:**
  - Start with a simple model first and then look at more complicated methods.
  - In non-linear systems, basic control methods (e.g., PID, LQR, Feedback Linearization) may perform satisfactorily, even in systems that are theoretically non-linear.
  - When using complex control strategies (e.g., Kalman Filters, Luenberger Observers), it is important to ensure precise tuning. Pay attention to things like observer gains and matrix precision to avoid performance issues.
  - Tailor controllers to your system and trajectory. Sometimes, methods that perform well on a specific reference trajectory may not generalize well to other cases.
  - Consider using a combination of controllers (e.g., LQG + PID) for flexibility in tuning different parts of the control system.
  - Always balance between performance (e.g., trajectory tracking) and system constraints (e.g., energy consumption).
- **Debugging and Problem Solving:**

- Pay attention to file version control and naming conventions to avoid confusion during testing and debugging. Use of GitHub is highly encouraged.
- Be mindful of hardware-specific issues that may arise, especially those that were not anticipated in the simulation.
- Use more advanced techniques, like the Rauch-Tung-Striebel smoother or Luenberger observers, if you encounter issues with noisy measurements or fluctuating estimations.
- Use trial-and-error or systematic tuning to determine good values for parameters like observer gains, LQR matrices, and controller poles.

- **Energy Consumption and Tracking Performance:**
  - A balance between energy consumption and tracking performance is crucial. Systems with higher energy consumption may perform poorly in terms of tracking error, while tight tracking may incur significant energy costs.

- **Learning from Simulation to Hardware Transition:**
  - During the transition from simulation to hardware, verify that parameters (e.g., torque constant, sampling period) match across different files to avoid mismatches.
  - Be ready to re-tune parameters when transitioning from simulation, especially if sensitive parameters are hardcoded or not consistently set.
  - Use simulations to fine-tune controllers, but be prepared for differences when moving to hardware.
  - Implement different noises in the simulation to achieve more robust hardware test performance.

- **Technical Suggestions:**
  - If you need to use the value of dt in your controller you will need to calculate this using the following equation rather than using the hardcoded value of 0.1.

$$dt = t - obj.t\_prev;$$

2 Feedback

Below are the feedback and lessons learned from the 2023 spring students. Lower scores are better.

- **Score: 6.30; Methods: LQG, Feedback Linearization** We learned that hardware testing is much more complicated than the simulation. We needed to be more careful in considering extra factors, such as the deadzone of the motor and the friction between the ball and the beam. While it is impossible to take all factors into account, these model mismatches lead to the delay or overshoot of the actual plant. As a consequence, the actual result performed worse than the simulated result. One of our main problems was the fluctuation of the estimation. Due to high amounts of noise in our position sensor, our estimator predicted larger values for the ball velocity than reality. We attempted to apply a moving average filter here as well, though it resulted in poor results that we were unable to debug. There are algorithms made specifically to smooth the output of a Kalman filter like the Rauch–Tung–Striebel smoother, which we would pursue instead if we were to continue this project. Adding a smoother might help to reduce the spikes in the controller command. We also might want to try a Luenberger observer instead of a Kalman filter, which drives the estimation error to zero by pole placement instead of identifying the appropriate uncertainty terms and variances. Some possible feedback may be having more instructions on the advantages/disadvantages of each controller regarding this particular system. Also, it would be nice to have more time available in the lab. This could be done by starting the lab component of the class earlier in the semester

- **Score: 2.50; Methods: Luenberger Observer, LQR, Feedback Linearization** There was little benefit to the angle going above/below 56 degrees. The controller wanted to send a large input at the beginning since theta started at -56 degrees, so within the first second, the input was saturated to 4 as both a physical safety limit as well as since little tracking benefit would have been gained. It was found that it was critical to start as close to initial conditions as possible on the beam (i.e., 0 position, 0 velocity) to match the reference trajectory for both LQR and feedback linearization. This makes sense since those control schemes linearize the system near the origin and so would perform best with an accurate starting position. When the angle of the beam was very small, and the ball was already stationary, static friction was observed to prevent the ball from moving, which was not modeled in the simulation. When tested on a reference sine wave trajectory, this would lead to flat peaks since the ball had a small velocity but could not reach the actual peaks due to friction. This was also seen when we tried to match a stationary reference trajectory, where the ball would move close and then stop a bit away even though the beam was slightly tilted. On the other hand, we found it difficult to fully reduce energy usage. In the feedback linearization, we cannot incorporate the energy cost into our design. In the LQR, while we could strike a balance between trajectory and energy cost, it was difficult to further reduce the overall cost. In the end, the greatest improvement to the score was tuning the poles and adjusting the biases and phase shifts to try and match the given reference trajectory. However it is noted that this method would tailor our controller to best

match a specific given reference trajectory, since this specific tuning may not work as well for different signals. Also, for trajectories that are not pre- defined, it may not be possible to look ahead in the reference signal and so there would be more lag in the tracking. A more general and robust controller could have been implemented, but it would still need to be tuned to better match the particular problem. Especially, control barrier function, Hamilton-Jacobian reachability, or Model Predictive Control might be used in order to explicitly consider the safety sets as hard constraints. There were some hardware specific issues that were not expected after running the simulations and so it was good to have two different controllers to use. Sometimes it was difficult to debug the root cause of the errors during the lab time so it was harder to implement more sophisticated solutions (e.x. Kalman filter), especially since they might also require debugging in hardware.

- **Score: 3.9; Methods: EKF, LQG+PID, Feedback Linearization** Overall the lab project was quite an interesting application of the non-linear control theory posed in class. The difference between simulation and real-life was expected, but aspects like static friction, un-modeled dynamics, and variances between machines required a significant amount of re-tuning to get right. The guarantees of overall compatibility with successful non-linear control strategies were thrown out the door due to the "messyness" of real world systems. A small overhaul had to be done to the LQG in order to track closer to the desired trajectory with a reasonable amount of energy consumption. Feedback linearization tuning was slightly intuitive and pushed us to develop other techniques or strategies to more accurately model the system we were working with. Tuning not only con- trollers but estimators meant a significant amount of balance between tool creation, parameter manipulation, and making sure estimators were behaving properly. If given more time I think our team would have focused on tuning parameters for improved performance. Overall though, our team was proud of the two strategies. The combination of LQG and PID proved to be a unique solution that allowed for flexible tuning of individual parts of our control system design. While the feedback linearization technique was more standard for the non-linear course, the integration of new state estimation and other components to combat uncertainty, proved a useful strategy for fine tuning behavior. For feedback the main "grievance" (if you could call it that) our team had was in the final scoring system. A heavy weight was placed against energy consumption versus the tracking error. Our team often observed very poor controllers, who did little to follow the trajectory, perform drastically better than controllers who were tight but had a larger energy impact. The LQG + PID controller had to be severely limited in energy output in order to score high. However, near perfect tracking could be achieved if the LQG + PID was permitted, but the energy penalty would shoot up to 5.0 - 7.0. While energy consumption makes this problem interesting, our team personally thought it was a bit skewed against penalizing energy consumption.

- **Score: 2.59; Methods: EKF, PID, LQI** In terms of hardware testing, we found it to be a valuable learning experience. We expected sensor noise to have a significant impact on the controllers' performance, but we did not expect the friction and modeling

errors to have as big of an impact as they did. This caused for a significantly lower performance than expected and highlighted the importance of accounting for factors such as friction and sensor noise in the simulation. Another big discovery we made, was in the choice of controller for the system. Early on, we attempted to implement Feedback Linearization and Backstepping controllers for the system, in accordance to what we had learned in class. However, this was deemed more challenging than expected, and the performance of the PID and LQI controllers, even though this system is non-linear, performed satisfactorily. If we were starting this lab again, we would definitely begin testing on the real system earlier in the process. This would have allowed us to identify and address any bugs earlier in the process, as well as more finely tune our gains to fit the real-life system. Overall, we have found this lab to be a valuable learning experience and would recommend it for future iterations. However, we would advise that future students be aware of the challenges involved in testing on physical systems and be prepared to adapt their control algorithms accordingly. An idea could be to introduce possibilities for noisy measurements in the simulations.

- **Score: 2.07; Methods: Approximate I/O, LQR, LQI** Our controller approximates a system without a well-defined relative degree by disregarding small non-linear terms, resulting in an output-linearizable system. After conducting tests on the hardware, it has been confirmed that this method is indeed feasible. Through this, we gained valuable knowledge on how to deal with similar systems when working on an actual engineering project. One thing that held us up for a while during the transition from simulation to hardware was making sure the parameters were the same across all the different files. While different than a model/plant mismatch issue, it is still an experienceand learning moment/experience. The issue we dealt with was not our controller parameters but rather system parameters (torque constant and sampling period) that were hardcoded differently between the simulation files and hardware testing files. Additionally, our teams pent awhile tuning parameters in simulation which over all helped us in testing hardware. But it is still an important balance to strike–spending time tuning in simulation while knowing that when transitioning the hardware the parameters (particularly ones that are sensitive) will need to be re-tuned/adjusted due to any mismatches between the simulation model and actual.

- **Score: 1.33; Methods: Feedback Linearization, LQR** Implementing the observer turned out to be one of the larger pain points. Using the matrix exponential and matrix integral to accurately discretize the de- veloped observer model worked quite well in MATLAB; expm executes just fine in the MATLAB environment. However when implementing this on the con- troller we had to hard-code the observer matrices. Initially we didn't include enough precision in the values we entered for the matrices, yielding unsatisfac- tory performance. However after settling on a set of observer gains we were happy with and using the precision afforded by vpa the hard-coded observer and LQR gains started working very well. As every software developer knows, version control is incredibly important when developing and debugging software. On the second lab day we ran into a bit of a snafu being unsure what file was being run and tested and whether our changes were being compiled onto the hardware. After solving this with

more judicious file naming and tracking our project really came together! After testing was completed, we came to the realization that we were quite aggressive during testing and tuning our gains. For instance, the project de- scription stated that the trajectory could have consisted of sinusoidal waves where amplitude varied between 0 to 15 cm and period varied between 6 to 10 seconds or square waves where the period variation was the same and the ampli- tude varied between 0 to 10 cm. Thus, we were testing our controllers with the most aggressive amplitudes and frequencies. This didn't seem to impact per- formance negatively when given the unknown reference trajectory, so this could have been one reason why our tracking worked relatively well. However, in the future, we should test a multitude of combinations of altitudes and periods–not just the most aggressive combination.

- **Score: 1.8; Methods: Luenberger Observer, I/O Linearization, LQR** For LQR control, we found a large difference between the simulated and hardware performances. We spent a lot of time tuning our LQR matrix in simulation, but had to re-tune our values once we tested on hardware. If we had to do this lab again, I think we would prioritize testing on hardware earlier to debug any issues that arose. I think compiling a list of common hardware problems could be helpful for future semesters. The takeaway is roughly the same with the IO controller. It worked more quickly in practice but tuning poles and gains was very crucial to make it work. It would be helpful to further develop our intuition about how to find good values for them, rather than by trial-and-error.

- **Score: 4.07; I/O Linearization, LQR** Hardware testing was more challenging than designing the controllers in MATLAB.The model/plant mismatch and the poor performance of our observer caused multiple issues such as poor tracking performance and high energy consumption. It was a drastic difference compared to the MATLAB simulation. We could have spent more time improving the performance of our controller. We stopped working on it as we obtained good scores in MATLAB simulations. In addition, we could have designed a second observer to be safer since our designed observer did not perform very well for the unobserved states. Furthermore, the I/O controller spent a massive amount of energy to control the system. We could have utilized the Control Lyapunov Functions to design a more efficient controller. This lab can be challenging for those without prior experience in control and hardware. It is our belief that incorporating more instruction on designing observers for real-life plants during lectures could lead to better outcomes, as opposed to a single instructive homework problem. Besides that instead of having two consecutive hardware testing, we can have lab sessions every other week so that we could have more time to debug and work on a different approach.

- **Score: ; LQR, Sliding-Mode Control** There were few lessons learned during this project: 1. The model of the system is a good representation of the system but it is not perfect. A controller design is an iterative process that starts at the simulation level, continues at the lab with the real system and then goes back for simulation updates until both systems behave close enough. 2. When tracking is important, always introduce an integrator to ensure zero steady-state error - even if in simulation

the steady state error is small. There are always unknown factors in the real world that influence the steady state error, and we have to actively ensure that it is small enough. 3. Always calibrate your system before starting measurements. At minimum, check that all measurements are reasonable and make sense.

- **Score: ; EKF, LQR, Feedback Linearization** During the tuning of the feedback linearization controller, we found that the control gains are tightly related to the performance of the system. The gain is not just a weight ratio for each state variable, but an indication of the system's stability. In the feedback linearization controller, the first control gain k1 determines the rise time of the system; k2 decides the height of the first peak; k3 is correlated with the settling time; k4 has effects on damping. The larger the k1, the system approaches the reference trajectory quicker. In the same manner, for a larger k2 and k3, the height of the first peak is lower and the time to settle is shorter. However, as k4 increases, the damping increases as well. This discovery might help the tuning of similar controllers which have periodic input and sinusoidal system state. Another lesson we learned is that the model/plant mismatch can be destructive to the system's stability. In our first run of the PD controller on the hardware, the system input grew to infinity after the first peak, which never happened in simulation. We assumed that this was caused by model mismatch. If we were to repeat the lab in the future, we would do many things the same way we did for our group. We would start with an estimator (probably an EKF, as it approximated the state remarkably well), and ensure this worked correctly. Then, we would prototype controllers, noting early on that a virtual input will likely be required (as we want to control the desired angle, which has hardware limits). For this reason, we would always work with two controllers–an inner and outer loop: one that sets the desired angle, and one that sets the desired motor input as a function of the desired angle. In terms of hardware, we would realize that model mismatch may cause instability, so we would ensure to set software limits for safety. In terms of feedback for the lab, we thought it was an interesting problem and a well-run project. One recommendation would be to give the groups (perhaps in the lab document) some of the insights from the previous paragraph (like the virtual input or model mismatch insights). Overall though, in terms of representing a real world problem with no constraints on solutions, the freedom was appreciated and helped us understand how to better approach this type of control project in the future.

3 Frequently Asked Questions

1. For the class project, we are given the function get_ref_traj.m. This function returns the position of the ball and its first two derivatives. However, following the procedure in the provided reference on approximate input-output linearization gives an approximate system with a relative degree of 4, so we would need two additional derivatives of the reference trajectory to implement the controller. Are we missing something? If not, can we modify get_ref_traj.m to include two additional derivatives?

   Answer: You are correct, we need these higher-order derivatives. Please numerically compute any higher-order derivatives you need rather than modifying get_ref_traj.m. We will be testing the controllers against an unknown reference trajectory where your controller will only have access to the reference position, velocity, and acceleration