31.5 out of 50. Well done

Control Design Project

Oliver Schamp

No proper abstract. Would be good to add background in abstract and introduction to introduce area/context and motivate need for benchmarking results that compare common controller tracking performance (covering basic, modern, and cutting-edge types of controllers). Remember to define acronyms

XTway. Part B -

I. Introduction

THIS design project for ELEC6228 Applied Control Systems has two parts. Part A deals with the mathematical modelling of a dual wheel inverted pendulum. PID and MPC techniques are applied to this model, the results of which are analysed and contrasted. Part B models rotary systems then applied in the physical domain using the Quanser Qube Servo. LQR, MPC and ILC are simulated and applied experimentally. These results are analysed, compared and contrasted.

II. A.1 Mathematical Modelling of the Robot Dynamics

Fig. 1: The Lego Mindstorms NXT Robot

The Lego Mindstorms NXT robot contains two motors which control its wheels on either side, along with a gyro sensor and an ultrasound sensor [1]. In this report, the robot is modelled as an inverted pendulum travelling along a single dimension, therefore a single input voltage is applied to both the left and right motors.

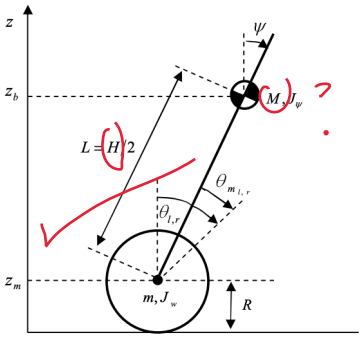


Fig. 2: Dual Wheel Inverted Pendulum Dynamics

The angle of rotation of the wheels is described by the coefficient θ, and the body pitch is described by the angle ψ. From robot, a equation $\frac{\delta^2\theta}{\delta t^2}$ and of each represen

Need full detail, i.e.
- define ss system. Define theta, psidot as measured variables. Define inputs
- mention [300Hz, 1kHz] zoh and [theta = pi/180, psi = pi/180] quantisation in Simulink to improve accuracy
- mention input is in PWM (0 – 100)
- mention noise and how much
- give schematic of Simulink system model with ZOH, saturation, noise, quantisation (not Simulink screen shot)
- mention non-zero i.c. and how much

III. Simulink

The state-space model was built in Simulink to analyse its properties and behaviour, along with a separate state-space system that was specified for the yaw angle ϕ of the robot also found in ELEC6228 coursework document.

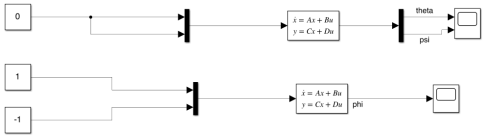
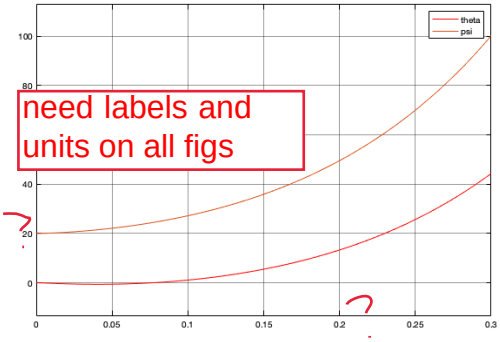


Fig. 3: NXT State-Space Systems in Simulink

When the state-space model containing θ and ψ was initialised at $X = [0\ \ 20\ \ 0\ \ 0]$ the robot falls forward in exponential fashion, reaching an angle that would correspond to the ground level (100 degrees) in 0.3 seconds. Furthermore, θ also increases, demonstrating that the momentum of the fall causes the robot to roll forward. This proves that several forces including gravity and friction, and other elements like the moment and inertia of the NXT have been modelled sensibly.



need labels and units on all figs

Fig. 4: NXT Falling Forward

Altho
state sp
voltages
increase

You don't describe any controllers until the QUBE section. The report would be clearer if you briefly state the PID update equation, the ss eqn and the LQT, MPC cost functions mathematically, so that signals and parameters you later discuss are defined more consistently and thoroughly throughout.

PID, which stands for Proportional, Integral, Derivative control is a method of designing a custom gain in a feedback

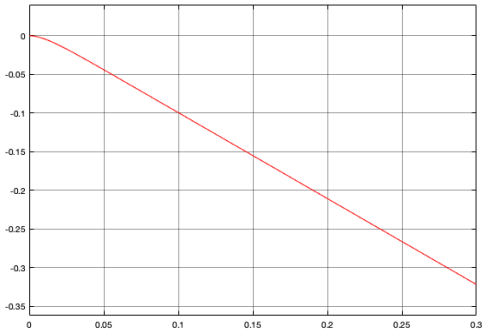


Fig. 5: ϕ with Opposite Voltages

control loop with three separate parts. The controller generates an output to be fed to the plant from an input describing the error of the system. The error is multiplied, integrated and differentiated so that the controller is sensitive to both values and changes in error. Each of these actions has a gain associated with it, which is tuned to best control the specific plant.

A single PID controller was first used to control the angle ψ, to stop the robot falling over. Simulink's PID blocks were used to tune the PID towards the best gain values automatically.

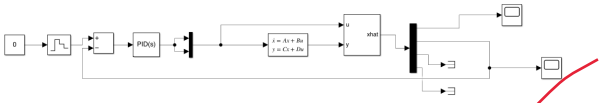
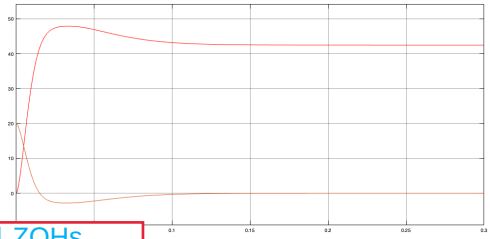


Fig. 6: PID Controlling ψ with a Reference 0

The tuning gave a proportional, integral and derivative gain of -181.7, -4802.0, -0.5 respectively. The controller stabilised the system in 0.2 seconds, rotating both wheels to an angle of around 50 degrees to keep the robot upright.



Stabilising from $\psi = 20$

did you add ZOHs, saturation, noise, quantisation block etc to make it as accurate as possible?

ntroller was then added in to control would return to its original position while balancing the inverted pendulum. The fact that the first controller was already tuned allowed the second controller to be tuned alone and create satisfactory performance, which was later improved by several tuning iterations back and forth between the controllers.

To test this extent of this performance, the reference for θ was set to a square wave while the reference for ϕ was set to 0, keeping the requirement of the robot balancing itself. The performance showed that PID is capable of this task when noise is not present.

Good – but usually a block diagram is clearer than a screenshot

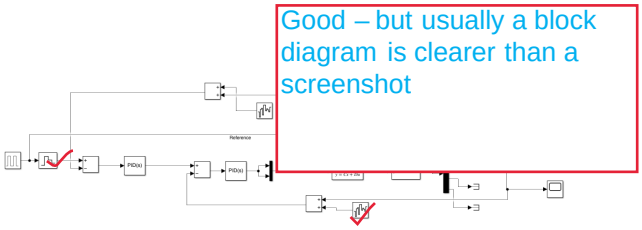
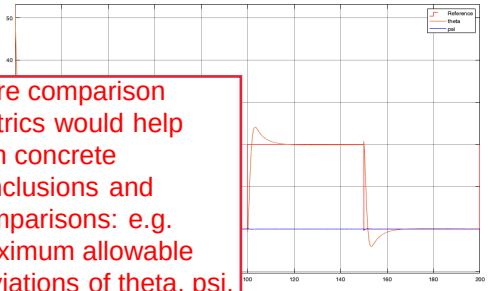


Fig. 8: PID Controlling θ via Square Wave Reference



More comparison metrics would help with concrete conclusions and comparisons: e.g. maximum allowable deviations of theta, psi, settling times, and input voltage

on Square Wave Reference

When noise is introduced to both PID controllers, θ handles noise power up to 90 before its variance from the maximum/minimum point of the reference surpasses the complimentary minimum/maximum reference value. For noise in ψ this level of uncertainty in θ is achieved at a noise power of 0.001, showing PID can be very sensitive to noise when controlling systems more complex and variable than simple rotating wheels.

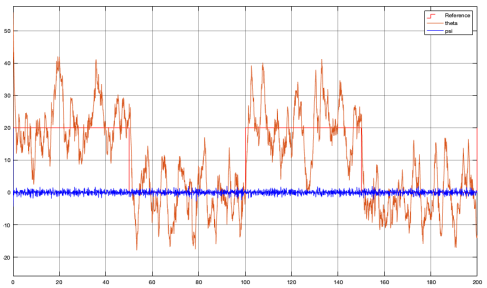


Fig. 10: PID Performance at θ Noise Power 90

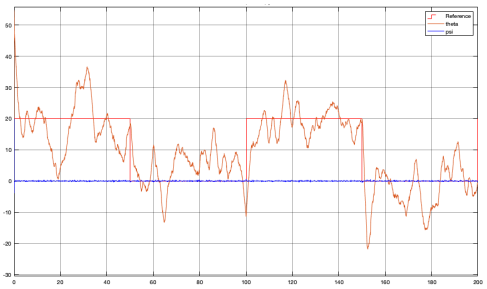


Fig. 11: PID Performance at ψ Power 0.001

V. A.4 Model Predictive Control

Model Predictive Control works by assuming that by modelling a finite horizon, you can generate the behaviour you need for your system to converge. By expanding the cost function equation 1 (assuming that we can express the current

state in the form of the input sequence $u_{t0}^{->}$ and a scalar through equation 2) it becomes a quadratic problem which can be minimised every control iteration for the optimum u_t.

$$J = \sum_{t=0}^{infinity} (x^T Q x + u^T R u) \quad (1)$$

$$x_{t+N} = A^N x_t + A^{N-1} B u_t ... + A^0 B u_t \quad (2)$$

must define all sigs and params

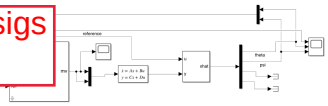


Fig. 12: MPC Controlling θ via Square Wave Reference

The MPC controller was designed using Simulink's MPC block. The controller was chosen using the 'MPC designer' tool, specifically modelled for square wave tracking. A prediction and control horizon of 15 worked best, with weightings of 1 and 0.5 for θ and ϕ to allow the MPC to track θ closely while also keeping ϕ contained.

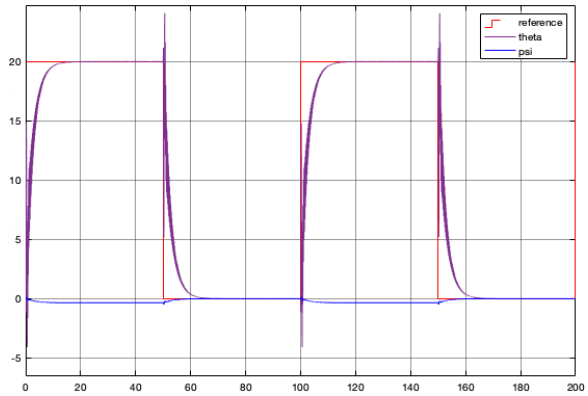


Fig. 13: MPC Performance on Square Wave Reference

If I were to use MPC explicitly without the MPC block, I would have to r by choosing the prediction horizon, eighting matrices R and Q used to g for the quadratic solver to find the optimum u_t. To tune these parameters, I would initialise an array with the time samples t. Then, using nested for loops for each of the previously mentioned parameters, I would generate an input vector from continually executing quadprog inside the nested for loops for each 't' sample. Using the 'lsim(u,t,ss-system)' function, I could generate responses for each of my customised systems, and then choose the parameters that elicit the best response. If needed, I could run this method again with refined parameters to optimise them further.

need professional journal style writing

Compared to the PID design in A.3, MPC fits the reference slightly better, and does not overshoot when a change in re However, the most notable thing ab C was that it handled noise much be ite noise power of 10 added to th imulation still gave very smooth results (albeit with an offset) therefore providing a notably more robust version of control. In figure 14, only if you look very closedly can you see asymmetries between the control attempts on the first and second periods of the reference.

did you use kalman? how did you tune. Again how did you you maximise model accuracy?

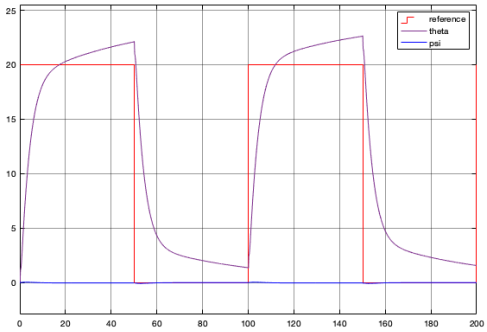


Fig. 14: MPC Performance on Square Wave Reference with Gaussian Noise Variance = 10

PID relies solely on the error between the reference and the output to generate the input for the next time step. MPC's quadratic solving method does not rely totally on the error of the output (expression 3, with the squared coefficient Q consisting of factors determined by the intrinsic nature of the state-space plant. This means that when MPC calculates the optimal input, it is much less likely to magnify wild errors caused by high noise magnitude due to the more indirect effect that noise and output error has on the control value generated.

$$min_x \frac{1}{2} x^T Q x + c^T x \quad (3)$$

VI. A.5 Discussion

In conclusion, either PID or MPC can be used to balance the Lego Mindstorms NXT robot. In a real system with feedback noise, an MPC solution would be preferred due to it performing much more robustly under noise. However the computing power needed to run MPC, where the matrix multiplication to set up the quadratic solver alone contains many more mathematical actions than a PID controller.

To improve my design in the case of PID, I would try controlling θ rather than ψ first before introducing the controller for θ. Perhaps this could give increased robustness for noise entering ϕ when θ must track a reference. To improve my design in the case of MPC, I would remodel it in an explicit form on Simulink, using the 'quadprog' equation and nested 'for' loops to iteratively try parameters. This would give me a lo sy th si w co ro as so comparatively much noisier performance.

Good to see basic controllers working. There are some param values given but would be nice to see more. Did you do much controller tuning? adding a range of metrics and more comparisons with different weights/ horizons etc would help find the trade-offs and justify your results. Must show effect of each param and show best performance had been achieved. Also need thorough (quantitative) comparison between controllers.

VII. B.1 QUBE Configuration

The QUBE Servo is controlled via LabVIEW running on the NI myRIO. The various control schemes were simulated in LabVIEW 2018, before implementation on the QUBE Servo for real-life experimentation. Quanser software was required to interface properly with the servo. Issues with LabVIEW add-ons hindered progress throughout the project. As such, some real-life experiments are saved for future work.

VIII. B.2 Mathematical Modelling

In this section, the mathematical modelling of the rotary pendulum is set out. Derivations are obtained from [2] A diagram of the pendulum and the associated parameters are given in Fig. 15. L_p is the length of the pendulum, the moment of inertia about the centre of mass of the pendulum is J_p. J_r is the moment of inertia of the rotary arm and the pendulum has an angle α. When the pendulum is in the upright position, $\alpha = 0$.

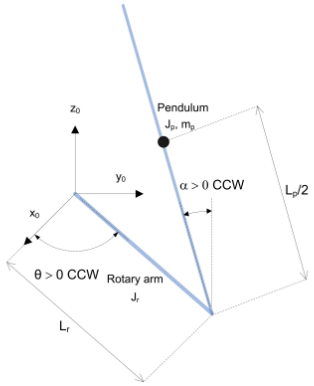


Fig. 15: Rotary Pendulum Diagram [2].

To establish the mathematical model, we use the Lagrange method as opposed to classical mechanics. In specific, the dynamics of the rotary arm and pendulum will be defined using the Euler-Lagrange equation, seen below:

$$\frac{\delta^2 L}{\delta t \delta \dot{q}_i} - \frac{\delta L}{\delta q_i} = Q_i \tag{4}$$

In this case the variable q_i is a generalized coordinate. Before we set out our equations of motion (EOMs), we define the following variable $q(t)$, and $\dot{q}(t)$ follows:

$$\begin{aligned} q(t)^T &= \begin{bmatrix} \theta(t) & \alpha(t) \end{bmatrix} \\ \dot{q}(t)^T &= \begin{bmatrix} \frac{\delta\theta(t)}{\delta t} & \frac{\delta\alpha(t)}{\delta t} \end{bmatrix} \end{aligned} \tag{5}$$

Where $\theta(t)$ and $\alpha(t)$ are the rotary arm and inverted pendulum angles respectively. QUBE Module parameters are given in Fig.16.

The Euler-Lagrange equations for the system are as follows:

Rotary Pendulum

m_r	Rotary arm mass	0.095 kg
L_r	Rotary arm length	0.085 m
m_p	Pendulum link mass	0.024 kg
L_p	Pendulum link length	0.129 m

Fig. 16: QUBE Servo 2 pendulum physical parameters

$$\begin{aligned} \frac{\delta^2 L}{\delta \theta^2} - \frac{\delta L}{\delta \theta} &= Q_1 \\ \frac{\delta^2 L}{\delta \alpha^2} - \frac{\delta L}{\delta \alpha} &= Q_2 \end{aligned} \tag{6}$$

Defining the Lagrangian $L = T - V$, and the forces acting on the rotary arm and pendulum, Q_1 and Q_2. The forces are expressed in terms of torque τ and damping B_p and B_r. In obtaining potential and kinetic energy and therefore the Lagrangian of the system, the nonlinear EOMs may be found.

$$\begin{aligned} Q_1 &= \tau - B_r \dot{\theta} \\ Q_2 &= -B_p \dot{\alpha} \end{aligned} \tag{7}$$

Hence the EOMs of the inverted pendulum is set out below:

$$\begin{aligned} &\left(m_p {L_r}^2 + \frac{1}{4} m_p {L_p}^2 - \frac{1}{4} m_p {L_p}^2 cos(\alpha)^2 + J_r\right)\ddot{\theta} \\ &-\left(\frac{1}{2} m_p L_p L_r cos(\alpha)\right)\ddot{\alpha} + \left(\frac{1}{2} m_p L_p^2 sin(\alpha) cos(\alpha)\right)\dot{\alpha}\dot{\theta} \\ &+\left(\frac{1}{2} m_p L_p L_r sin(\alpha)\right)\dot{\alpha}^2 = \tau - B_r \dot{\theta} \\ &-\frac{1}{2} m_p L_p L_r cos(\alpha)\ddot{\theta} + \left(J_p + \frac{1}{4} m_p {L_p}^2\right)\ddot{\alpha} \\ &\qquad sin(\alpha) = -B_p \dot{\alpha} \end{aligned} \tag{8}$$

Well done for tackling pendulum. Good detail (although full derivation could just be referenced)

system, we need to translate the problem into the state-space domain in order to apply the various control techniques. By linearisation of the EOMs, and calculating the moments of inertia of the pendulum link and rotary arm, we can express the system in the form of a state-space equation, as seen here:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{9}$$

Where $x^T = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]$ is the state, and $y^T = [x_1, x_2]$ is the output. Values of C and D can be easily identified, as only servo position and link angles are dependent variables. Therefore C and D matrices are:

$$\begin{aligned} C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \\ D &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \tag{10}$$

By substitution of the state into our EOMs, the A and B matrices can then be found. J_T is the total inertia of the link (J_p) and rotary arm (J_r).

$$A = \frac{1}{J_T}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A1 & A2 & A3 \\ 0 & A4 & A5 & A6 \end{bmatrix}$$
$$B = \frac{1}{J_T}\begin{bmatrix} 0 \\ 0 \\ J_p + \frac{1}{4}m_p {L_p}^2 \\ \frac{1}{2}m_p L_p L_r \end{bmatrix} \quad (11)$$

Where:

$$A1 = \frac{1}{4}{m_p}^2 {L_p}^2 L_r g$$
$$A2 = -(J_p + \frac{1}{4}m_p {L_p}^2)B_r$$
$$A3 = -\frac{1}{2}m_p L_p L_r B_p \quad (12)$$
$$A4 = \frac{1}{2}m_p L_p g(J_r + m_p {L_r}^2)$$
$$A5 = \frac{1}{2}m_p L_p L_r B_r$$
$$A6 = -(J_r + m_p {L_r}^2)B_p$$

Again perhaps full ss params could be referenced (especially if you didn't id them)

Now that we have established our state-space equation, and defined the state, input, output and feed-forward matrices (A, B,C and D respectively) according to the EOMs we can niques. nverted

It is a shame you didn't try model identification and validation. This is a key part of model-based control and would improve later results.

IX. B.3 LQR Optimal Control

A. Optimal Control Theory

In order to achieve optimal control, an optimal input must be selected such that a cost function, commonly referred to as J, is minimized. Where J takes the following form:

$$J = \sum_{t=0}^{\infty} \left({x_t}^T Q x_t + {u_t}^T R u_t \right) \quad (13)$$

To obtain our optimal input sequence ${u_t}^*$, we solve the tion (DARE) and define

Would be useful to define and apply consistent performance objectives (e.g. error norm, max error pend, combined norm, max input Vm, t_theta, t_alpha post disturbance) to thoroughly assess and compare performance

$$B^T PB)^{-1} B^T PA$$
$$B^T PA \quad (14)$$

As such, the optimal input sequence is obtained and the system is controlled. Now we can proceed to apply the optimal control method to the case in question, the QUBE inverted pendulum.

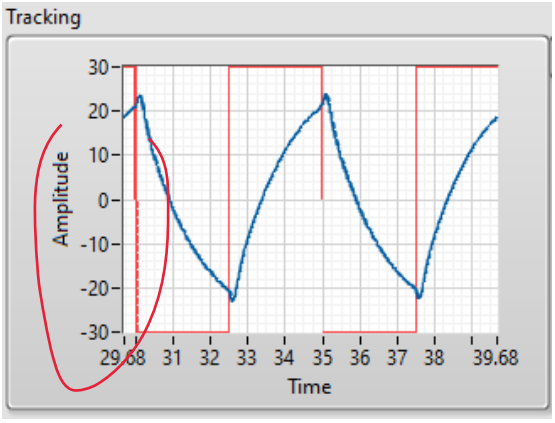


Fig. 17: Plot of physical servo angle and reference under default conditions.

B. Application via LabVIEW

A block diagram of the LQR controller can be seen in Appendix B. When subject to external input, we expect that the control system works to balance the pendulum by way of matching the desired reference. Hence the task may be considered as a tracking problem.

Optimal control via LQR requires proper tuning of Q and R values to achieve stabilisation of the system. To assess the effect of a properly tuned LQR controller, the device is first simulated using the default paramete a previous VI using the Labview LQR bloc 0.2Hz, initial amplitude is $\alpha = 30^o$, and the is $[-1, 30, -1.75, 3]$. The resulting tracking is Fig. 17.

You can't mean 0.2Hz! This is far too slow

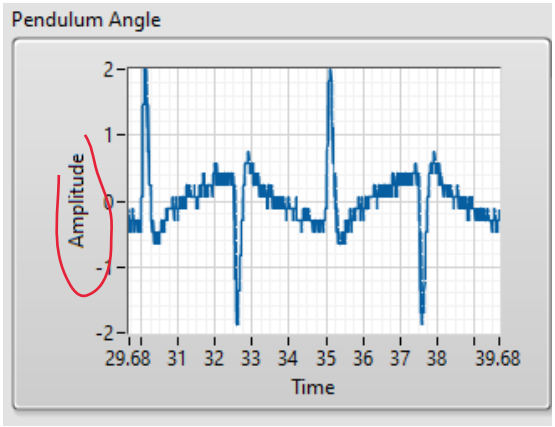


Fig. 18: Physical pendulum angle over time under default conditions.

As we may expect, the system is weak in tracking the reference signal, and the average deviation of the pendulum angle $\alpha_d ev$ is 25.7^o. Fig. 18 shows the pendulum angle over time. While the average angle is near the balancing point, the pendulum is oscillating significantly. Hence we require tuning of the Q and R values, and therefore the controller gain K, to achieve a more robustly balanced system.

Following trial and error testing, we arrive at the following tuned gain K, where $K = [-6.5, 36, -2, 3.4]$. As before the

frequency is 0.2Hz and the initial angle $\alpha_0 = 30^o$. Upon simulation of the system, the following results are obtained.

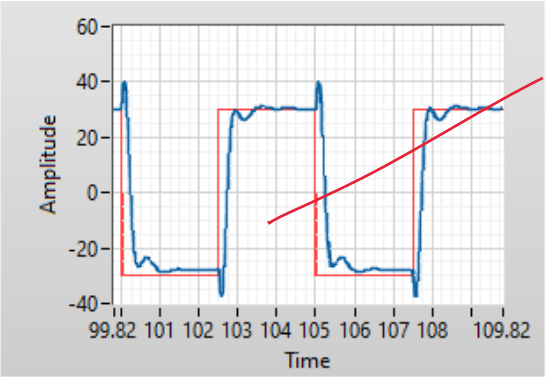


Fig. 19: Plot of servo angle over time with tuned LQR controller for a real system.

Fig. 19 shows far more accurate tracking of the reference signal. While the servo angle experiences spikes during a change in the reference, the controller acts swiftly to approach the desired servo angle. Furthermore, the pendulum angle is significantly more stable over time, as seen in Fig. 20. The average deviation from the balancing point has been reduced to 8.6^o.

Would be good to see more results for different tuning parameters against several metrics to thoroughly assess and justify choice of parameters used and the trade-offs involved

Fig. 20: Pendulum angle over time using tuned LQR controller for a real system.

X. B.4 Model Predictive Control

A. MPC Theory

While the LQR controller achieves balancing of the pendulum, we would like to increase the robustness of the balancing via a more complex control scheme, such as MPC. MPC is popular in the industry thanks to the ability to overcome feedback delay. While LQR implements the optimal solution for the whole time window, MPC performs optimization in a receding horizon, theoretically allowing for a better solution. As before we need to select suitable Q and R values in order to minimise the performance index J, as shown in the equation 15.

$$J = \sum_{t=t_0}^{t_0+N-1} \left({x_t}^T Q x_t + {u_t}^T R u_t \right) \tag{15}$$

Firstly, we use the state and system model to produce our prediction of the system output, expressed as:

$$\vec{x}_{t_0} = G\vec{u}_{t_0} + Hx_{t_0} \tag{16}$$

Expanding the finite horizon cost function using equation 16 gives a quadratic to be minimised:

$$min_x \frac{1}{2} x^T Q' x + c^T x \tag{17}$$

Then, as before, find our optimal input sequence, denoted as $u^*_{t_0}$, implement the control input, and then repeat for the receding horizon. Where:

$$u^*_{t_0} = f(x_{t_0}) \tag{18}$$

Subject to the following constraints:

$$\begin{gathered} |u_t| \leq M, t_0 \leq t < t_0 + N \\ \begin{bmatrix} -M \\ : \\ -M \end{bmatrix} \leq \begin{bmatrix} u_{t_0} \\ : \\ u_{t_0+N-1} \end{bmatrix} \leq \begin{bmatrix} M \\ : \\ M \end{bmatrix} \longrightarrow \\ \longrightarrow \begin{bmatrix} I \\ -I \end{bmatrix} \vec{u}_{t_0} \leq \begin{bmatrix} \vec{M} \\ \vec{M} \end{bmatrix} \longrightarrow \\ \longrightarrow F\vec{u}_{t_0} \leq b \end{gathered} \tag{19}$$

B. Application via LabVIEW

A block diagram of the MPC controller in LabVIEW can be seen in Appendix C. In our implementation of MPC, we used Labview's built-in blocks to provide a faster performance to support the longer control horizon needed to optimise a complex system such as a rotary pendulum. Before implementation of our control scheme onto the QUBE servo, we establish an unconstrained baseline model in simulation for comparison with the experimental results. The simulated model uses a prediction horizon of 11, an initial angle of 30, and a sampling time of 0.1 seconds. A horizon of 11 was chosen after it was found using Simulink tha required to achieve any functi are obtained.

are these the best results? Did you try other weights and use performance measures to compare and validate your tuning?

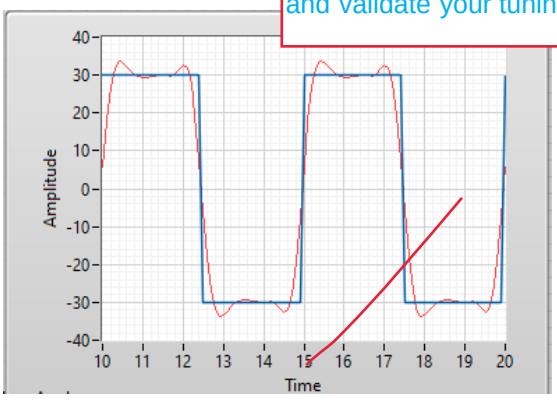


Fig. 21: Plot of servo position versus reference over time for an MPC simulation.

The simulated control scheme produces strong tracking of the reference signal, which is further demonstrated by observing the sharp yet controlled changes in the pendulum angle over time.

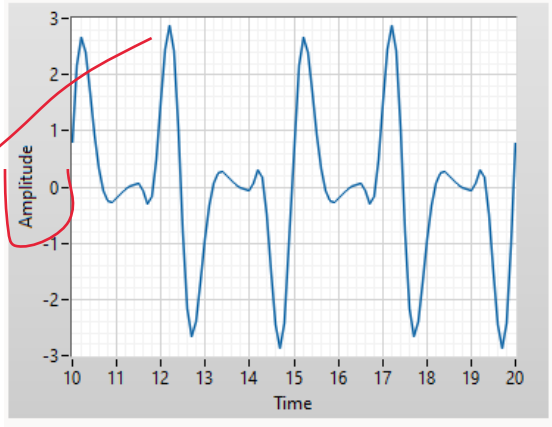


Fig. 22: Pendulum angle over time in simulated MPC scheme.

In the simulation, the MPC controller produces a more accurate balancing performance compared to LQR, with an average deviation of 4.6^o from the balancing point. The corresponding servo voltage is shown below in Fig. 23. The low control voltage along with a small deviation from the balancing point suggests a more robust balancing operation compared with previous solutions. Hence the motivation for a higher complexity yet more robust controller is demonstrated.

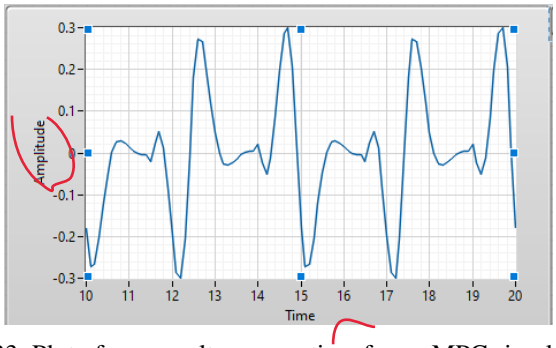


Fig. 23: Plot of servo voltage over time for an MPC simulation.

The following results show the deterioration of performance in simulation when constraints are introduced. In this case, the servo voltage is bounded between ± 0.08.

From Figs. 24 and 25, we can see a reduced level of performance in balancing the pendulum. Furthermore, the average deviation from the balancing point is increased to 11.8^o.

Now we aim to implement the control scheme onto the QUBE servo and obtain real-life results.

Firstly, we tested the MPC constraints on the real voltage input:

Secondly, we balanced the pendulum using unconstrained MPC. To obtain our best performance (Figure 27b) the prediction horizon was increased to 120 and the control horizon was increased to 20. The sampling time was decreased to 0.005s, and the values along the diagonals of the Q and R matrices

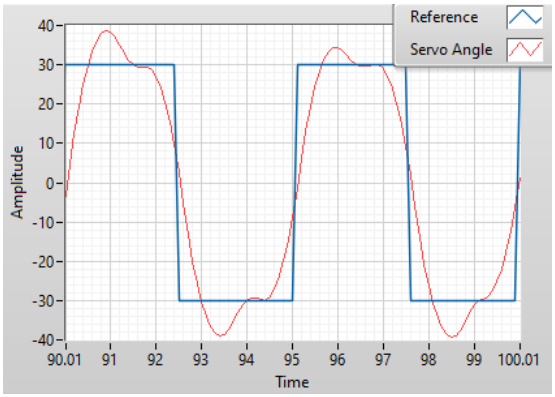


Fig. 24: Plot of servo angle versus reference for constrained MPC simulation.

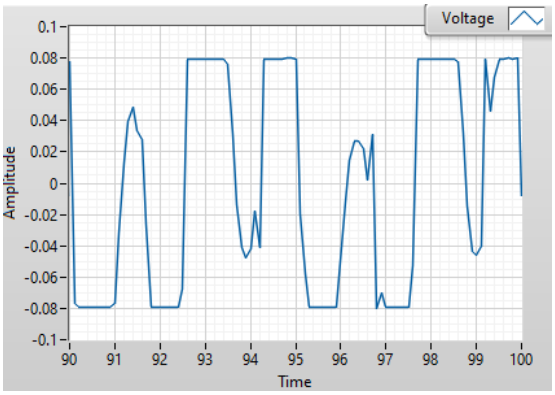


Fig. 25: Plot of servo voltage over time for constrained MPC simulation.

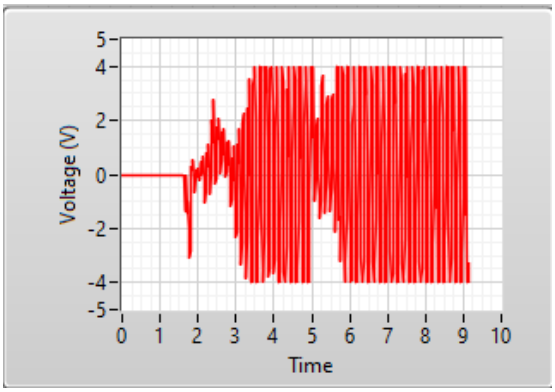


Fig. 26: Plot of servo voltage constrained over time for on Qube Servo.

were set to magnitudes of 7.5 and 5 to provide more weight on the output error when minimising the MPC quadratic.

Compared with our simulated results, the QUBE servo requires a significantly higher motor voltage in order to achieve stability. Figure 27b provides better tracking than the LQR, at an average deviation from the reference of 6.2^o. We observed that the more performant the MPC became, the greater the variance in voltage to the Servo. This is explained by the increased output error weightings c to the control action change weightings $Q^{'}$ (equation 17) ratio of $\frac{7.5}{5}$ after tuning.

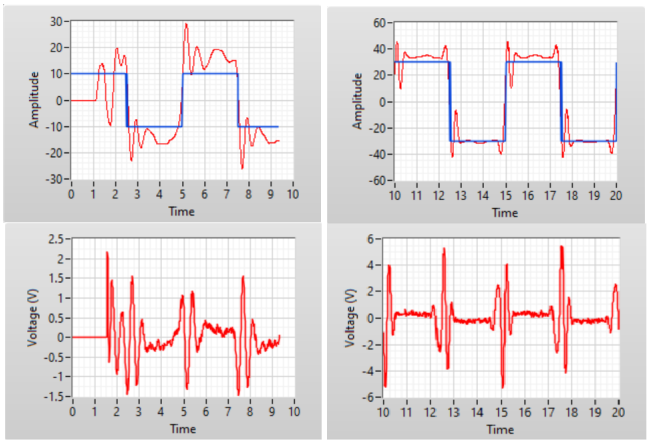


Fig. 27: Comparison between the voltage and tracking of: a) Untuned MPC controller and b) Tuned MPC controller on Qube Servo

again, more comparisons would be helpful, compared against several metrics to find effect of changing each parameter. Need to convince reader the highest accuracy has been achieved

nstrained design is by no means the best tion and finer tuning of the system to reduce be a topic of future experiments, because not work as well with constraints applied.

ITERATIVE LEARNING CONTROL

efined using the well established state-space equation, this time using k as the number of iterations, as shown:

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{20}$$

In this way, the system tracks a given reference iteratively over a finite time interval. For the given system model, and using the lifted form modelling, we obtain the tracking error, e_k:

$$\begin{aligned} y_k &= Gu_k + d \\ e_k &= r - Gu_k - d \end{aligned} \tag{21}$$

Such that the following cost is minimised:

$$||e_{k+1}||^2 + \omega\gamma_k^2 \tag{22}$$

Where ω is a weighting parameter and γ_k is the learning gain.

B. Application via LabVIEW

The following results show the effect of changes to the weighting parameter on the system control.

Tracking plots for $\omega = 1/1e15$ and $\omega = 1/1e100$ were also recorded, and produce near identical results to that of $\omega = 1e5$. Therefore the error norm is also identical, as seen in Fig. 30. The controller reaches an accurate copy of the reference after 1000 iterations, and an even more accurate value after 2000 iterations, assuming $\omega \leq 1e5$.

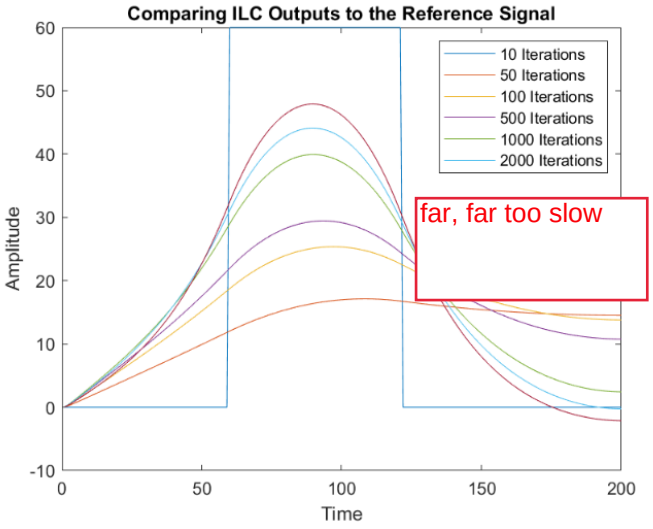


Fig. 28: ILC simulation output versus reference signal for $\omega = 1e15$.

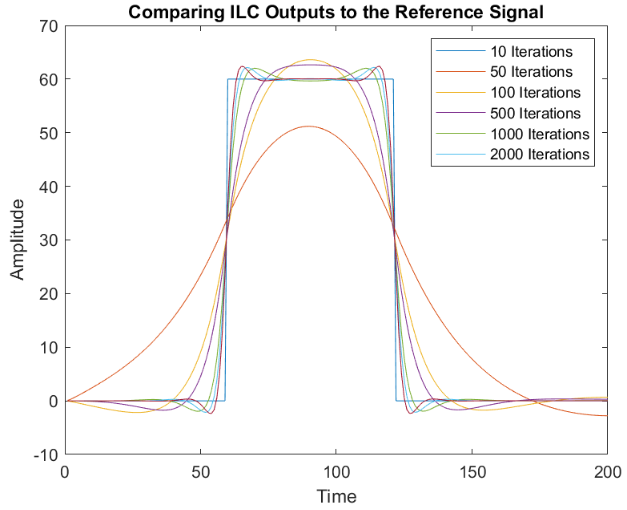


Fig. 29: ILC simulation output versus reference signal for $\omega = 1e5$.

What a shame ILC was not implemented. Perhaps applying NOILC would have been simpler (just one update eqn)

the
ex
controller. As such, these experiments are saved for future work.

XII. DISCUSSION

nothing for part A?

In I onstrated as a good quality control method by which the pendulum is balanced. As before, we would like to improve robustness, hence the MPC scheme was introduced. MPC provided more robust balancing with a lower deviation from the centre point, allowing the servo to track the reference more closely. However, due to the number of parameters involved, our final implementation of MPC on the Qube can still be improved. We were able to constrain the input voltage to the Qube with MPC, but only managed to balance the pendulum with constraints in simulation. The

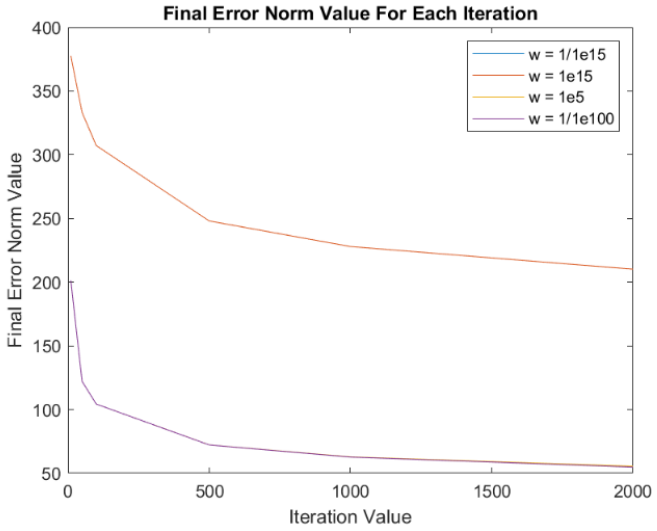


Fig. 30: Plot of final error norm value for each iteration in simulation.

challenges we faced when applying MPC to the Qube Servo were higher and more erratic motor voltages than expected, which made debugging difficult as the voltage would often exceed axis limits so we could not observe the exact behaviour of the input. Future work on MPC and LQR would include further finer tuning of the sensitive control parameters.

Simulation of ILC also showed promising results. Providing a suitable weighting parameter allowed for an accurate reproduction of the reference signal after 1000 iterations, and an error converging to zero. As a result of the delays caused by the software add-ons, it was decided that the ILC controller would be implemented on the DC servo motor as it was simpler and would not take as long, but we were not able to implement this on the QUBE. In the future, we would aim to implement this control process onto the QUBE servo and then tackle the pendulum problem using an ILC controller. The approach that would be taken is to first balance the pendulum using LQR after which the ILC will be used to track the reference signal which would result in a balanced pendulum tracking a reference signal.

REFERENCES

[1] E. A. C. Systems, "Applied control systems: Design project," 2022.
[2] A. Chang, "Quanser rotary pendulum workbook, instructor verson," 2011. [Online]. Available: https://www.quanser.com/products/qube-servo-2/

Impressive that you tackled the pendulum problem. Well done for getting MPC and LQR working quite well. Use of simulations prior to experiments was thorough. Conclusions were general and made sense. A few minor comments:

- Would be good to see more comparisons between paras/tuning weights to justify that you have optimised the tuning. I'd like to see the compromises/trade-offs you made.
- Adding more performance metrics would have expanded the validation and added more thorough justification (i.e. quantitative via tables)
- Some model id tests and more thorough tests to validate model would be useful
- Would be good to include constraints on all controllers to enable deeper comparisons
- Descriptions of controllers sometimes unclear (i.e. use different notation to previous sections) – perhaps better to provide controller descriptions at start of report and use more consistently to ensure all parameters are clear and control structure is well explained
- Figs could be tidied up, and controller descriptions made more consistent.
- It is a great shame ILC was not implemented

APPENDIX A
DIFFICULTIES WITH LABVIEW INSTALLATION

To configure the QUBE, LabVIEW 2018 was installed from iSolutions onto our computers. The 'MyRIO Bundle', including the Mathscript RT library, the Control and Simulation library and the Real-time library for LabVIEW was then also installed onto our personal computers. Following this was the Quanser software which provided the LabVIEW blocks necessary to communicate with the QUBE (all the downloads happened on personal computers). iSolutions had originally supplied the LabVIEW add-ons separately on their website, but the Real-time module was for LabVIEW 2018 SP1 while all other files were compatible with LabVIEW 2018. This caused many difficulties in troubleshooting an issue that was impossible to solve, leaving less time to test, tune and optimise our control systems.

APPENDIX B
MPC BLOCK DIAGRAM

APPENDIX C
LQR BLOCK DIAGRAM

APPENDIX D
ILC BLOCK DIAGRAM

APPENDIX E
PARAMETERS TUNING FRONT END

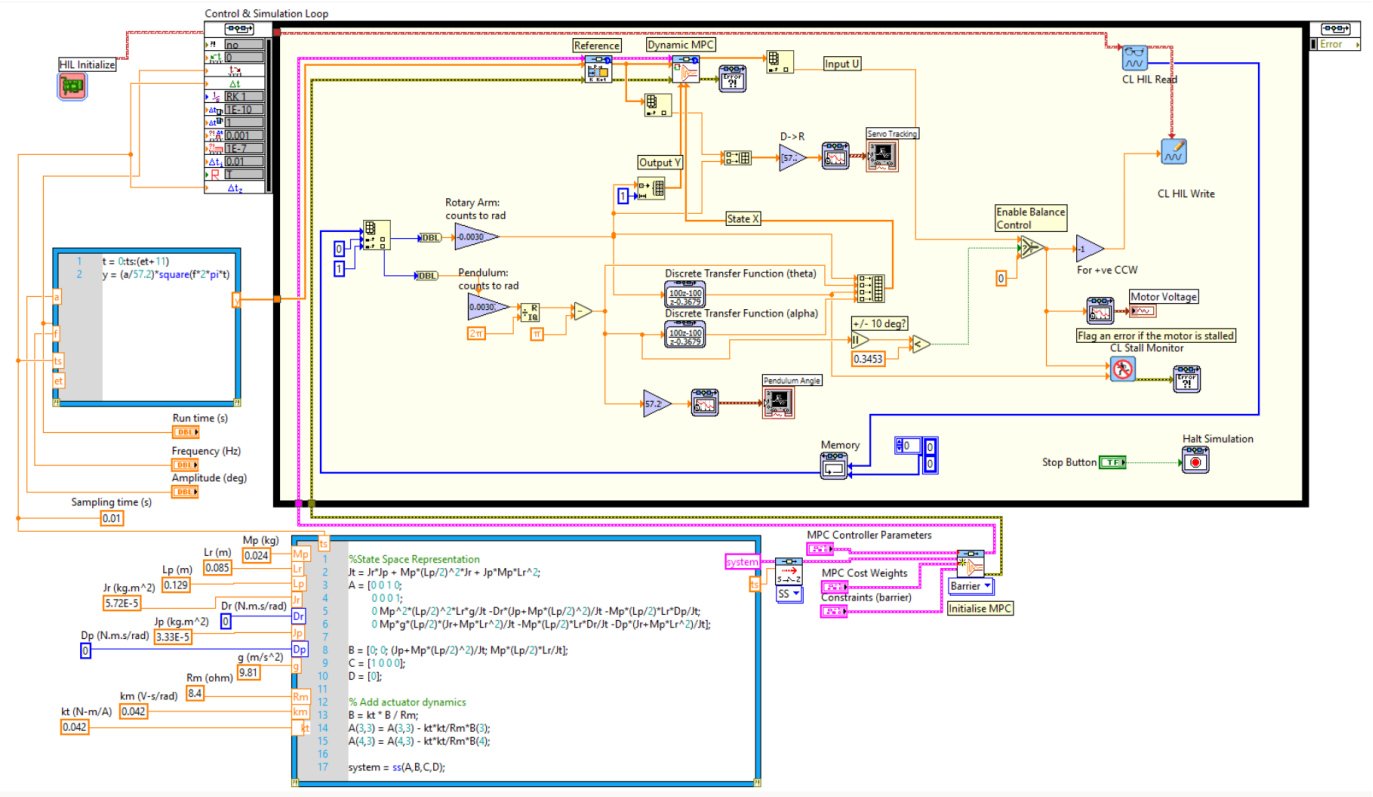


Fig. 31: MPC Qube Servo Block Diagram (Pendulum)

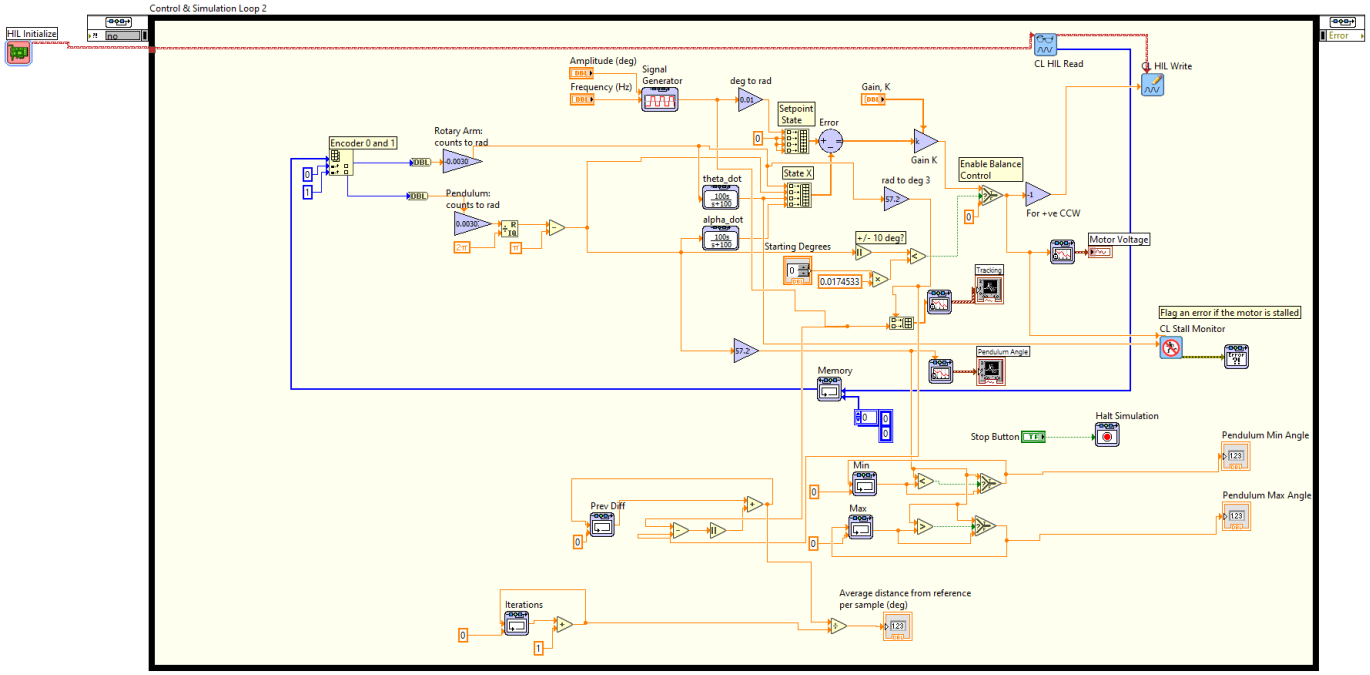


Fig. 32: LQR Qube Servo Block Diagram (Pendulum)

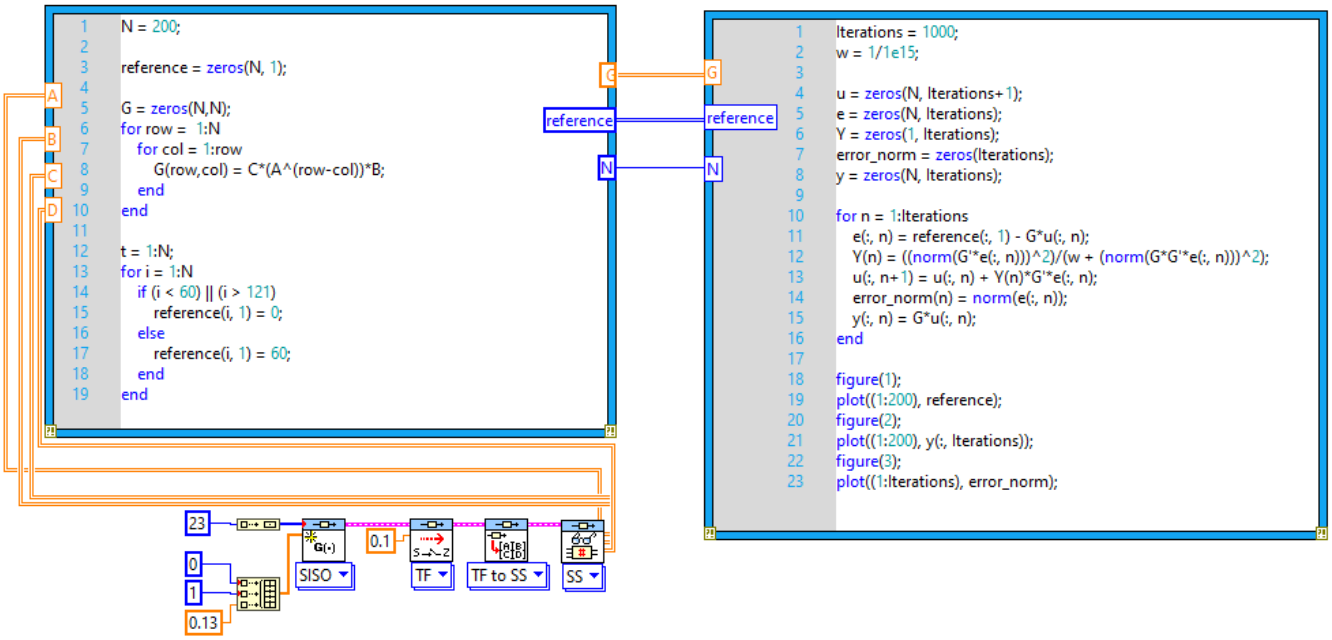


Fig. 33: ILC Qube Servo Block Diagram (Inertia Disk)

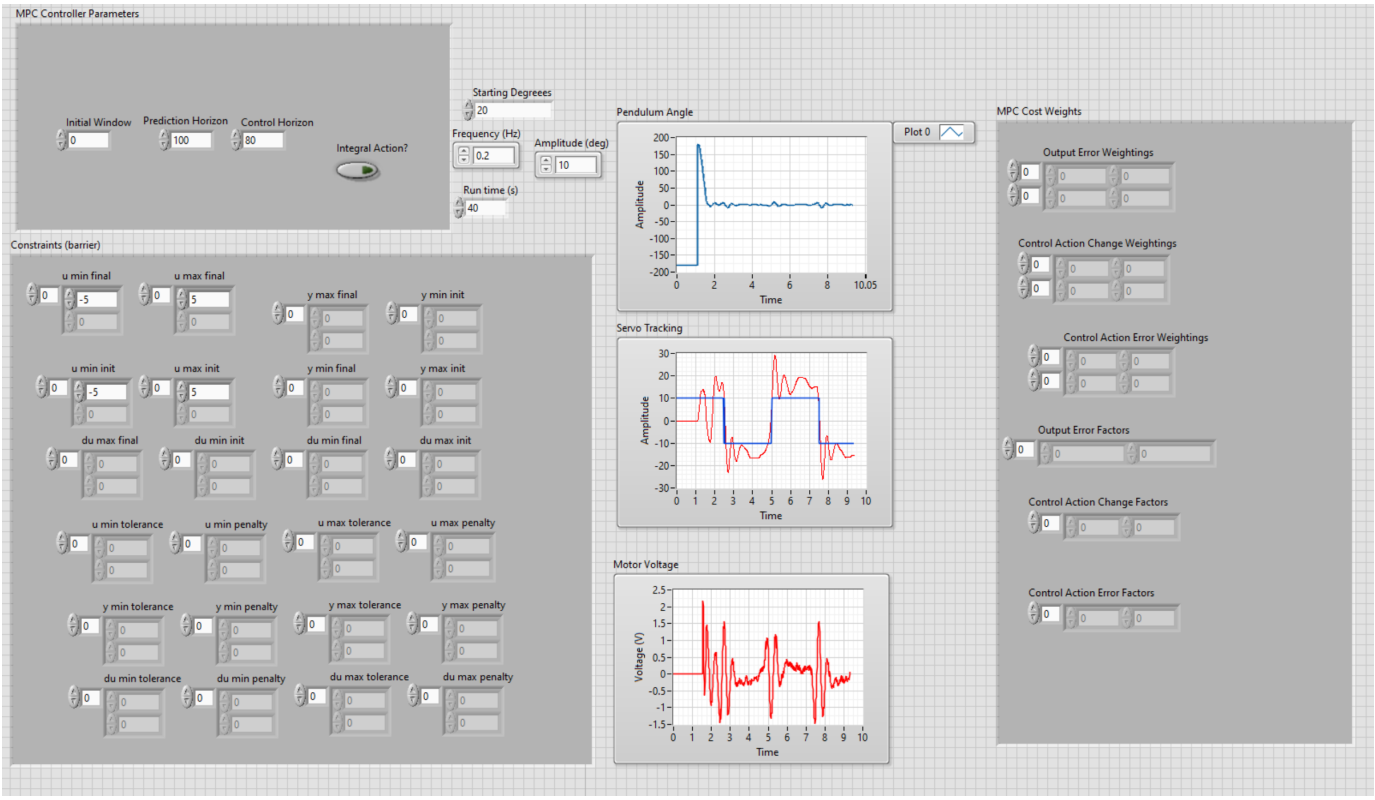


Fig. 34: Physical implementation of an untuned MPC.

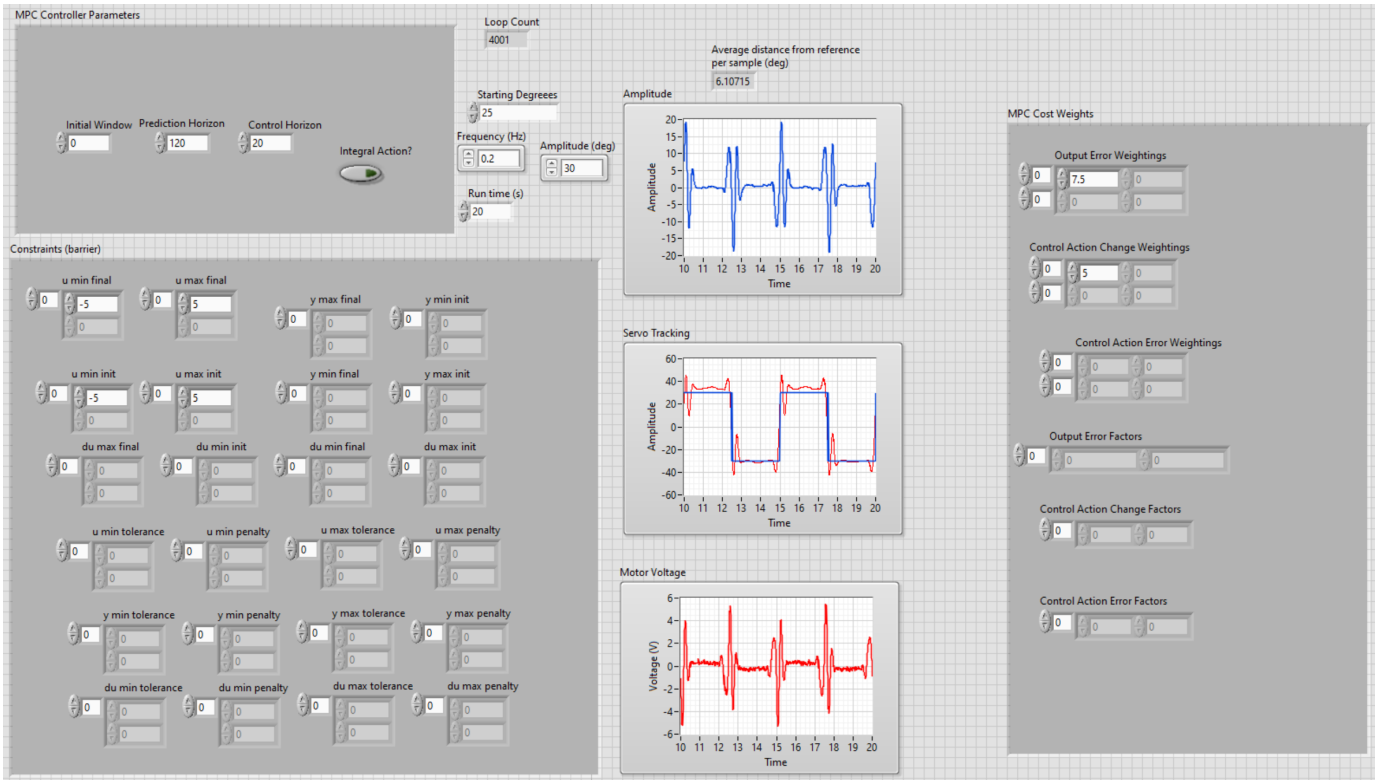


Fig. 35: Physical implementation of a tuned MPC.