AE 567: Project 4 - The Coupling of Model Predictive Control with Robust Kalman Filtering

Anirudh Aatresh[1]

***Abstract*— Building an approximate model that describes certain phenomenon or systems is a common practice. Apart from other characteristics, two models that describe a certain system can differ on how accurate they are in doing so and how much computation they require to do so. Ideally, we would want to have a very accurate model that also requires less computation to run. This model uncertainty can affect controllers such as model predictive control (MPC) as they rely on a model of the system to obtain a control input at every time step. In this project, we look at how we can implement a more robust state estimator for the MPC framework that takes into account the fact that the model we are working with is only an approximation of the system. We apply these controllers to a tracking problem on a servo-mechanism system.**

I. Background

Model predictive control (MPC) is a very popular algorithm for constrained control tasks. It is a standard due to the robustness of the control solution that it provides and the simplicity of the solution. One particular advantage that it provides is that it is receding horizon control, which means that the control solution at a particular instant is determined by considering the system output and input at the current time step and a fixed number of subsequent time steps. The MPC formulation is based on a model of the system which we are trying to control. Often, this model is not exactly known and only an approximation of this model can be estimated through system identification means. The quality of this approximation determines the quality of the control solution in the tracking problem. A very good quality model is able to give more accurate relationships between the control input and the system output. However, real world systems are often non-linear in nature and require complex non-linear functions in the model description. Processing these non-linear relationships places an additional computational load on the hardware and makes using complex models in real-time systems slow and difficult. Ideally, we would want to build the simplest model that also accurately describes the system.

Linear model are simple approximations that are often used to describe a system. This model, Σ, can be described mathematically as:

$$\begin{aligned} x_{k+1} = &\quad Ax_k + Bu_k + Gv_k \\ y_k = &\qquad Cx_k + Dv_k \end{aligned} \tag{1}$$

where, $x_k \in R^{n\times 1}$, $u_k \in R^{q\times 1}$, $y_k \in R^{p\times 1}$, $A \in R^{n\times n}$, $B \in R^{n\times q}$, and $C \in R^{p\times n}$. $v_k \in R^{n\times 1}$ is additive noise that is combined with the state and observation through matrices G and D, such that $G^T D = 0$. Matrices G and D here are the square roots of the covariance matrices of the state process noise and observation noise.

In a tracking problem, let us assume that r_k is the trend the output of the system must follow at each time step. In the MPC formulation, we would like to minimize the following cost function at every time step k, $J_k(u_k, \Sigma)$,

$$\begin{aligned} J_k(u_k,\Sigma) = &\quad \textstyle\sum_{i=1}^{H_p} ||r_{k+i} - \hat{y}_{k+i|k}||^2_{Q_i} + \\ &\textstyle\sum_{j=0}^{H_u-1} ||\hat{u}_{k+j|k} - \hat{u}_{k+j-1|k}||^2_{R_i} \end{aligned} \tag{2}$$

where, H_p and H_u are the prediction and control horizons respectively, $\hat{y}_{k+i}$ is the output prediction at time step $k+i$ of y_{k+i}, $\hat{u}_{k+j}$ is the control prediction at time step $k+j$ of u_{k+j} and $u_k = [u_{k|k}, ..., u_{k+H_u-1|k}]^T$. Q_i is the weight associated with the output prediction at time step $k+i$ and R_j is the weight associated with the control variation at $k+j$. Following this formulation, the optimal control at time step k is

$$u_{k|k} = \text{argmin}_{u_k}(J_k(u_k, \Sigma)) \tag{3}$$

In the unconstrained MPC formulation, there exists a closed form solution to this problem and is

$$u_{k|k} = [I_q\, 0 \ldots 0](H^{-1}G) \tag{4}$$

where $H = \Theta^T Q \Theta + R$ and $G = \Theta^T Q(r_k - \Psi\hat{x}_{k|k})$. The matrices used here are:

$$\Psi = \begin{bmatrix} (CA)^T \\ (CA^2)^T \\ \ldots \\ (CA^{H_p})^T \end{bmatrix} \tag{5}$$

$$\Theta = \begin{bmatrix} CB & 0 & \ldots & 0 \\ CAB & CB & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ CA^{H_p-1}B & \ldots & \ldots & CA^{H_p-H_u}B \end{bmatrix} \tag{6}$$

$$Q = \text{diag}(Q_1, Q_2, ..., Q_{H_p}) \tag{7}$$

$$R = \text{diag}(R_0, ..., R_{H_u-1}) \tag{8}$$

$$r_k = [r_{k+1}, r_{k+2}, ..., r_{k+H_p}]^T \tag{9}$$

where $\hat{x}_{k|k}$ is an estimate of x_k at time step k. This estimate is usually determined using a Kalman filter. In this project, I have considered the simpler case of the unconstrained MPC formulation over the more complex constrained MPC formulation due to the ease of implementation and the close form solution that exists for the optimal control. The approach for robust Kalman filtering that has been explained and implemented in subsequent sections are also applicable to the constrained MPC case, as mentioned in [1]. The

1 Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, USA, `aaatresh@umich.edu`.

constrained MPC does not have a closed form solution and an iterative approach can be taken.

A. Kalman Filtering

The Kalman filter is a state estimator for solving the filtering equations when dealing with linear dynamics and a Gaussian approximation of the state distributions. The state space model can be described as:

$$\begin{aligned} x_{k+1} = &\quad Ax_k + Bu_k + \xi_k \\ y_k = &\quad\quad Cx_k + \eta_k \end{aligned} \tag{10}$$

where ξ_k is the state process noise and η_k is the noise in the observation model. These noises can be expressed in terms of their distributions as $\xi_k \sim \mathcal{N}(0, GG^T)$ and $\eta_k \sim \mathcal{N}(0, DD^T)$

The generalized prediction equation in Gaussian filtering can be expressed as:

$$P(X_k|\mathcal{Y}_{k-1}) = \int P(X_k|X_{k-1})P(X_{k-1}|\mathcal{Y}_{k-1})dX_{k-1} \tag{11}$$

The generalized update equation for Gaussian filtering can be expressed as:

$$P(X_k|y_k, Y_{k-1} = \{y_0, ..., y_{k-1}\}) = \frac{P(y_k|X_k)P(X_k|Y_{k-1})}{P(y_k|Y_{k-1})} \tag{12}$$

When dealing with linear dynamics, we can simplify these equations to obtain the Kalman filtering prediction equations as:

$$m_k^- = Am_{k-1} + Bu_{k-1} \tag{13}$$

$$P_k^- = AP_{k-1}A^T + GG^T \tag{14}$$

Subsequently, we can obtain the update equations associated with the Kalman filter as:

$$m_k = m_k^- + P_k^- C^T (CP_k^- C^T + DD^T)^{-1}(y_k - Cm_k^-) \tag{15}$$

$$P_k = P_k^- - P_k^- C^T (CP_k^- C^T + DD^T)^{-1} CP_k^- \tag{16}$$

In the standard MPC formulation, at every time step a new optimal control is determined through the closed form expression shown in equation 4. Next, the same dynamics is forward propagated to determine the output. However, the next state of the system is estimated through a Kalman filtering prediction step using the observation that is obtained. In [1], these Kalman filtering equations have been expressed slightly differently in notation as follows.

Firstly, let us define a few utility matrices:

$$L_k = P_k C^T (CP_k C^T + DD^T) \tag{17}$$

$$K_k = AL_k \tag{18}$$

$$P_{k+1} = AP_k A^T - K_k (CP_k C^T) K_k^T + GG^T \tag{19}$$

Let us then look at the update equation:

$$\hat{x}_{k|k} = x_{k|k-1} + L_k (y_k - C\hat{x}_{k|k-1}) \tag{20}$$

where, $\hat{x}_{k|k}$ is the updated mean.

The prediction equation can be directly written from the previous predicted state mean as:

$$\hat{x}_{k+1|k} = Ax_{k|k-1} + K_k (y_k - C\hat{x}_{k|k-1}) + Bu_k \tag{21}$$

where, $\hat{x}_{k+1|k}$ is the prediction mean.

This Kalman filter formulation can be used in a standard MPC formulation to determine estimates of the next state at each time step. The standard MPC law has bee described in algorithm 1. In this law, we start off with an observation and initial state that is drawn from a Gaussian distribution that is centered around zero and has covariance matrix P_0. Next, for each time step, the optimal control is obtained using the updated estimate of the state at that time step, which is provided by the Kalman filter.

Algorithm 1: Standard MPC Law

```
Data: $K$ : Number of time steps for which
      simulation must be run.
Result: Sequence of optimal control inputs as a
        function of time step $U_{1:K}$, sequence of
        system outputs as a function of time step
        $Y_{1:K}$, $Y_0 = y_0$.
$k \leftarrow 0$ /* Time step counter */ ;
$y_0 \leftarrow 0$ ;
$P_0 = Var[\xi]$ ;
$x_{0|-1} \sim \mathcal{N}(0, P_0)$ ;
while $k < K$ do
    $L_k = P_k C^T (CP_k C^T + DD^T)$
    $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - C\hat{x}_{k|k-1})$
    $H = \Theta^T Q \Theta + R$
    $G = \Theta^T Q (r_k - \Psi \hat{x}_{k|k})$
    $u_{k|k} = [I_q\, 0 \dots 0](H^{-1}G)$
    $U_k = u_{k|k}$
    /*Applying $u_{k|k}$ to the system*/
    $x_{k+1|k} = f(\hat{x}_{k|k}, u_{k|k}, k)$
    $y_{k+1} = h(x_{k+1|k}, k)$
    $Y_{k+1} = y_{k+1}$
    $K_k = AL_k$
    $P_{k+1} =$
      $AP_k A^T - K_k (CP_k C^T + DD^T) K_k^T + GG^T$
    $\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) + Bu_k$
    $k \leftarrow k + 1$
end
```

B. Servo-mechanical System

To demonstrate my work in this project, I use the application demonstrated in [1] and [2], the servo-mechanical system. A servo is a mechanical actuator consisting of a motor, gearbox and load. This can be seen in figure 1. This actuator finds widespread application in robotics due to the flexibility provided in usage. The underlying non-linear

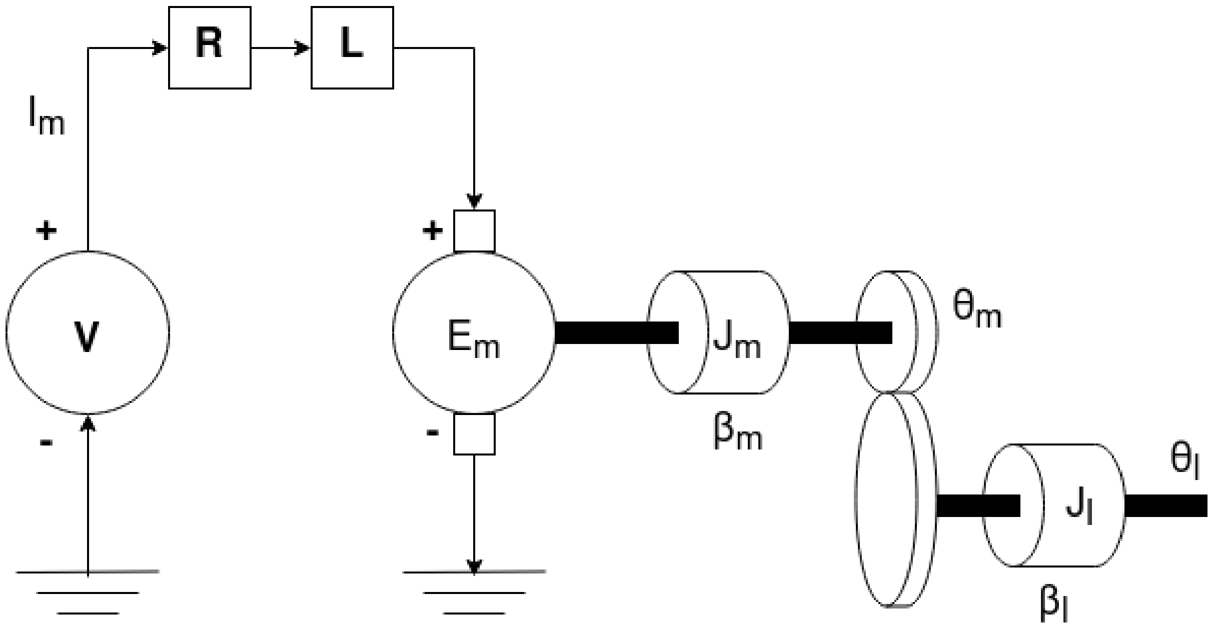


Fig. 1: Servo-mechanical system

model describing this system is:

$$J_l\ddot{\theta}_l = \rho T_s - \beta_l\dot{\theta}_l - T_{fl}(\dot{\theta}_l) \tag{22}$$
$$J_m\ddot{\theta}_m = T_m - T_s - \beta_m\dot{\theta}_m - T_{fm}(\dot{\theta}_m) \tag{23}$$
$$T_m = K_t I_m \tag{24}$$
$$V = RI_m + L\dot{I_m} + E_m \tag{25}$$
$$E_m = K_t\dot{\theta}_m \tag{26}$$
$$I_m = \frac{(V-E_m)}{R}(1-e^{\frac{-Rt}{L}}) \tag{27}$$
$$T_s = \frac{k_\theta}{\rho}(\frac{\theta_m}{\rho} - \theta_l) \tag{28}$$

The non-linearities in the state update equations comes from the following equations:

$$T_{fl}(\dot{\theta}_l) = (\alpha_{l_0} + \alpha_{l_1}e^{-\alpha_{l_2}|\dot{\theta}_l|})\text{sgn}(\dot{\theta}_l) \tag{29}$$
$$T_{fm}(\dot{\theta_m}) = (\alpha_{m_0} + \alpha_{m_1}e^{-\alpha_{m_2}|\dot{\theta_m}|})\text{sgn}(\dot{\theta}_m) \tag{30}$$

and sgn(x) is defined as:

$$\text{sgn}(x) = \begin{Bmatrix} 1 & x>0 \\ 0 & x=0 \\ -1 & x<0 \end{Bmatrix} \tag{31}$$

The variables and constants in the non-linear dynamics are shown in table I. Here, the values that were suggested in [1] and that were used in my experiments have been shown for these parameters. It must be noted that here, ϵ_{max} and ϵ_{min} are used as parameter perturbations on the linear model (nominal model) described in the next paragraph. This is because the values of the parameters of the non-linear model are difficult to estimate and need not be necessarily accurate. Those parameters that whose nominal value is considered reliable enough are perturbed with ϵ_{min} and otherwise are perturbed with ϵ_{max}.

The linear version of these dynamics is obtained by removing the non-linear terms T_{fl} and T_{fm}, and assigning L to 0. These linear dynamics can be expressed as:

$$J_l\ddot{\theta}_l = \rho T_s - \beta_l\dot{\theta}_l \tag{32}$$
$$J_m\ddot{\theta}_m = T_m - T_s - \beta_m\dot{\theta}_m \tag{33}$$

The parameter values of the nominal model can be seen in table II. More information about this servo-mechanical system and the nominal parameters values can be found in [2].

Let us consider the state of this system to be $x_k = [\theta_{l,k},\, \dot{\theta}_{l,k},\, \theta_{m,k},\, \dot{\theta}_{m,k}]^T$. The continuous time non-linear dynamics were discretized using the zero-order hold method and sampling time ΔT. This can be expressed as:

$$x_{k+1} = x_k + \Delta T f(x_k, u_k, k) \tag{34}$$

The continuous time linear dynamics were discretized in the following manner:

$$A = e^{A_c\Delta T} \tag{35}$$
$$B = A_c^{-1}(e^{A_c\Delta T} - I)B_c \tag{36}$$
$$x_{k+1} = Ax_k + Bu_k \tag{37}$$
$$y_k = Cx_k \tag{38}$$

where A_c and B_c are the continuous time linear system matrices obtained from equations (32) and (33).

It must be noted that the input must be constrained to make this simulation as realistic as possible. Considering [2] as a

TABLE I: Real parameters of the Servo-mechanical system

Symbol	Meaning	Value
ϵ_{min}	Minimum model perturbation	0.05
ϵ_{max}	Maximum model perturbation	0.1
L	Inductance of armature coil	0.8
J_m	Moment of inertia of the motor	$0.5(1+\epsilon_{\text{max}})$
β_m	Motor viscous friction coefficient	$0.1(1+\epsilon_{\text{max}})$
R	Resistance of armature coil	$20(1+\epsilon_{\text{min}})$
K_t	Motor constant	$10(1+\epsilon_{\text{max}})$
ρ	Gear ratio	$20(1+\epsilon_{\text{min}})$
k_θ	Torsional rigidity	$1280.2(1+\epsilon_{\text{min}})$
J_l	Moment of inertia of the load	$25(1+\epsilon_{\text{max}})$
β_l	Load viscous friction coefficient	$25(1+\epsilon_{\text{max}})$
$[\alpha_{l_0}, \alpha_{l_1}, \alpha_{l_2}]$	Load non-linear friction parameters	[0.5, 10. 0.5]
$[\alpha_{m_0}, \alpha_{m_1}, \alpha_{m_2}]$	Motor non-linear friction parameters	[0.1, 2, 0.5]

TABLE II: Nominal parameters of the servo-mechanical system

Symbol	Meaning	Value
L	Inductance of armature coil	0
J_m	Moment of inertia of the motor	0.5
β_m	Motor viscous friction coefficient	0.1
R	Resistance of armature coil	20
K_t	Motor constant	10
ρ	Gear ratio	20
k_θ	Torsional rigidity	1280.2
J_l	Moment of inertia of the load	25
β_l	Load viscous friction coefficient	25

guide for these dynamics, the following constaint is applied on the input at every time step:

$$|u_k| \leq 220V \tag{39}$$

II. Problem Setup

When building controllers for real world systems, we must work with the non-linear relationships between system states and inputs. However, building and implementing a non-linear model that accurately describes the system is computationally expensive to run on hardware. Instead, we would prefer simple linear models that are much cheaper in computational complexity. However, this linear model is only an approximation of the system we are trying to control and this approximation leads to sub-optimal solutions in the control input for the tracking problem using a standard MPC. Instead of using the standard Kalman filter in the MPC, we instead follow the suggestion of [1] to use the robust Kalman filter [3], that performs state estimation with the knowledge that the model we are working with is only an approximation of the true model. This controller is much more robust towards approximation errors and in tackling model uncertainty. To compare the two controllers, we implement them for a tracking problem with the servo-mechanism described in section I-B.

III. Proposed Methods

The standard Kalman filtering formulation is vulnerable to modelling errors. These modelling uncertainties play a big role in causing deviations in good state estimates. Kassam and Poor [4] - [5] did early work on optimal filters that helped mitigate these problems by solving a minimax problem. In this approach, a set of possible system models are considered to lie in the neighborhood centered about a nominal model. In the minimax optimization approach, an optimal filter is created for the least favorable model. This early work although promising, had the issue of being very difficult to implement. However, more recent work in this area has lead to the revival of the minimax viewpoint, in the context where modelling errors are described as norms of state-space dynamics perturbations. Levy et al in [3] follows this path by proposing the re-interpretation of risk sensitive filters for its use in this minimax viewpoint.

The risk sensitive filtering strategy is based on replacing the standard quadratic loss function in the least squares filtering problem with an exponential quadratic loss function, which is used to heavily penalize large errors. In the context of using risk sensitive filters, model uncertainties are described as an N-D ball of possible models around the nominal model. The radius of the ball is considered to be a tolerance parameter and the distance measure here is the relative relative entropy between a possible model and the nominal one.

Let us consider the nominal model to be Σ and the actual model to be $\tilde{\Sigma}$. The minimax solution towards obtaining the optimal control at time step k is a modification of equation (3):

$$u_{k|k} = \text{argmin}_{u_k} \text{max}_{\tilde{\Sigma} \in S} J_k(u_k, \tilde{\Sigma}) \tag{40}$$

The robust MPC formulation is based on solving this optimization problem at the step where the optimal control is obtained. Robust MPC has previously shown good results because of the fact that it considers the possibility that we may not know the exact model description and takes into account this model uncertainty when determining the optimal control. More details about this kind of thought process incorporation in the MPC framework has been described in subsequent paragraphs.

Previous work has focused on solving this minimax problem. However, the computation required to solve equation (40) is more demanding than that of solving a minimization problem. Due to this, the authors in [1] instead propose implementing robust MPC in two separate steps. Firstly, use a robust Kalman filter to perform robust state estimation under model uncertainties, and second, use the standard MPC formulation mentioned in algorithm 1.

Suppose that our linear model Σ is of the form in equation (1). Let the noise samples v_k be independent and identically distributed. Let us assume that the initial state, x_0, is drawn from a Gaussian distribution: $f_0(x_0) \sim \mathcal{N}(\bar{x}_0, \bar{P}_0)$. Let us create a new random vector $z_k = [x_{k+1},\ y_k]^T$ which is a combination of the next state and current observation. At time step k, the model Σ is completely described by the conditional probability of z_k given the measurements $Y_{k-1} = [y_0, ..., y_{k-1}]^T$, which is denoted by $f_k(z_k|Y_{k-1})$. As x_{k+1} and y_k are normally distributed, their joint distribution ($f_k(z_k|Y_{k-1})$) will also be Gaussian. Now, let us consider the actual conditional probability distribution to be $\tilde{f}_k(z_k|Y_{k-1})$. Similar to the approach of risk sensitive filtering, we would like to use KL divergence as a distance metric to determine the closeness of the nominal model and the actual model. The KL divergence is a way to measure the statistical distance between two distributions and compare their similarity. The KL divergence between distributions f_k and $\tilde{f}_k$ can be described as:

$$\mathcal{D}(\tilde{f}_k, f_k) = \int \tilde{f}_k(z_k, Y_{k-1})\ln\left(\frac{\tilde{f}_k(z_k|Y_{k-1})}{f_k(z_k|Y_{k-1})}\right) dz_k \quad (41)$$

Using this measure, we can now define the set of all allowable models S_k to be:

$$S_k = \{\tilde{f}_k(z_k|Y_{k-1})|\mathcal{D}(\hat{f}_k, f_k) \leq c\} \quad (42)$$

where c is the tolerance threshold. Set S_k denotes all possible models lying inside a ball of radius c centered at f_k.

Let us define a robust state estimator, g, as:

$$\hat{x} = g(y) \quad (43)$$

Now, we can define the robust state estimate as an estimate x_{k+1} given Y_k as a solution of the following minimax problem

$$x_{k+1|k} = \text{argmin}_{g_k}\ \max E_{\hat{f}_k}[||x_{k+1} - g_k(y_k)||^2|Y_{k-1}] \quad (44)$$

I have assumed that f_k is normally distributed, $f_k \sim \mathcal{N}(m_z, K_z)$. It can be shown that $\tilde{f}_k$ is also normally distributed [3], that is, $\tilde{f}_k \sim \mathcal{N}(\tilde{m}_z, \tilde{K}_z)$. Their covariance matrices can be written as:

$$K_z = \begin{bmatrix} K_x & K_{xy} \\ K_{yx} & K_y \end{bmatrix};\ \tilde{K}_z = \begin{bmatrix} \tilde{K}_x & K_{xy} \\ K_{yx} & K_y \end{bmatrix} \quad (45)$$

where the cross and co-variance matrices, K_{xy} and K_y, are not perturbed and only K_x is perturbed to $\tilde{K}_x$ [3]. If we define P and $\tilde{P}$ to be the error covariance matrices,

$$P = K_x - K_{xy}K_y^{-1}K_{yx} \quad (46)$$
$$\tilde{P} = \tilde{K}_x - K_{xy}K_y^{-1}K_{yx} \quad (47)$$
$$\tilde{P} = (P^{-1} - \tau I)^{-1} \quad (48)$$

where $\frac{1}{\tau}$ is a Lagrange multiplier corresponding to the constraint $\mathcal{D}(\tilde{f}_k, f_k) \leq c$ in the optimization problem in equation (44). The value of $\frac{1}{\tau}$ is determined by choosing its value that ensures the Karush-Kuhn-Tucker (KKT) condition holds:

$$\frac{1}{\tau}(c - \mathcal{D}(\tilde{f}_k, f_k)) = 0 \quad (49)$$

If we try to evaluate the expression for the KL divergence, mentioned in equation (41), it can be simplified through the Gaussiantity of the input distributions to:

$$\mathcal{D}(\tilde{f}_k, f_k) = \frac{1}{2}\left[\text{tr}(\tilde{P}P^{-1} - I) - \ln(\det(\tilde{P}P^{-1}))\right] \quad (50)$$
$$= \frac{1}{2}\left[\text{tr}((I - \tau P)^{-1} - I) + \ln(\det(I - \tau P))\right] \quad (51)$$

From these formulations, we can define the value of τ as the solution to:

$$\gamma(\tau) = \text{tr}((I - \tau P)^{-1} - I) + \ln(\det(I - \tau P)) - c \quad (52)$$

Algorithm 2: Robust MPC Law

```
Data: $K$ : Number of time steps for which
      simulation must be run.
Result: Sequence of optimal control inputs as a
        function of time step $U_{1:K}$, sequence of
        system outputs as a function of time step
        $Y_{1:K}$, $Y_0 = y_0$.
$k \leftarrow 0$ /* Time step counter */ ;
$y_0 \leftarrow 0$ ;
$P_0 = Var[\xi]$ ;
$x_{0|-1} \sim \mathcal{N}(0, P_0)$ ;
while $k < K$ do
    Find $\tau_k$ as the solution of
     $\ln(det(I - \tau_k P_k)) + \text{tr}[(I - \tau_k P_k)^{-1} - I] = c$
    $\tilde{P}_k = (P_k^{-1} - \tau_k I)^{-1}$
    $L_k = \tilde{P}_k C^T(C\tilde{P}_k C^T + DD^T)$
    $\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - C\hat{x}_{k|k-1})$
    $H = \Theta^T Q \Theta + R$
    $G = \Theta^T Q(r_k - \Psi\hat{x}_{k|k})$
    $u_{k|k} = [I_q\, 0 \ldots 0](H^{-1}G)$
    $U_k = u_{k|k}$
    /*Applying $u_{k|k}$ to the system*/
    $x_{k+1|k} = f(\hat{x}_{k|k}, u_{k|k}, k)$
    $y_{k+1} = h(x_{k+1|k}, k)$
    $Y_{k+1} = y_{k+1}$
    $K_k = AL_k$
    $P_{k+1} =$
     $A\tilde{P}_k A^T - K_k(C\tilde{P}_k C^T + DD^T)K_k^T + GG^T$
    $\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) + Bu_k$
    $k \leftarrow k + 1$
end
```

The new covariance matrix $\tilde{P}$ can be used to create a more robust form of the standard Kalman filtering formulation. The robust Kalman filter prediction and update equations respectively become:

Algorithm 3: Bisection algorithm

```
Result: $\tau_k = \tau_1$
$\tau_1 \leftarrow \epsilon$;
$\tau_2 \leftarrow \lambda - \epsilon$    where $\lambda$ is the largest eigenvalue of $P_k$;
 $\gamma(\tau) = \ln(det(I - \tau P_k)) + \mathrm{tr}[(I - \tau P_k)^{-1} - I] - c$
 while $|\tau_1 - \tau_2| \geq \epsilon$ do
     $\tau_{\text{new}} = \frac{\tau_1 + \tau_2}{2}$
     if $\gamma(\tau_{new}) < 0$ then
         $\tau_1 \leftarrow \tau_{\text{new}}$;
     else
         $\tau_2 \leftarrow \tau_{\text{new}}$;
     end
end
```

$$\tilde{P}_k = (P^{-1} - \tau_k I)^{-1} \tag{53}$$

$$L_k = \tilde{P}_k C^T (C\tilde{P}_k C^T + DD^T) \tag{54}$$

$$K_k = AL_k \tag{55}$$

$$P_{k+1} = A\tilde{P}_k A^T - K_k (C\tilde{P}_k C^T) K_k^T + GG^T \tag{56}$$

$$\hat{x}_{k|k} = x_{k|k-1} + L_k (y_k - C\hat{x}_{k|k-1}) \tag{57}$$

$$\hat{x}_{k+1|k} = Axk|k-1 + K_k (y_k - C\hat{x}_{k|k-1}) + Bu_k \tag{58}$$

Algorithm 2 shows the use of the robust Kalman filter to find the optimal control in the standard MPC framework. The function $\gamma(\tau)$ has been shown in [1] to be monotonically increasing and convex in τ. Hence, to solve this root finding problem, I have used the simple bisection algorithm, which was also proposed in [1]. The bisection algorithm used in my implementation can be found in algorithm 3.

IV. Results

The controllers and dynamics were implemented in Python and the code can be found in the appendix. Two sets of experiments were conducted. First, I consider the scenario of when the actual model and the nominal model coincide. This means that at every time step, the forward propagation of the dynamics corresponds to forward propagating the nominal model. The second scenario corresponds to having the actual model and the nominal model not coincide and that the nominal model is only an approximation of the actual one. This means that at every time step, the forward propagation corresponds to working with the actual non-linear model.

In both experiments, I have considered a tracking problem where the load angle must reach a set point of $r_k = \frac{\pi}{2}$rad. A sampling time of $0.1s$ has been used here for the discretization. In experiment 1, that is, where the nominal model coincides with the actual model, a system process noise and observation noise of standard deviation 0.03 (3 % noise was assumed to realistic here) was added. The dimension of the state space was $n = 4$, dimension of control input space was $q = 1$, and the dimension of the outpout space was $p = 1$. Both experiments were simulated for 20 seconds. Moreover, in both scenarios, the prediction horizon and control horizon were set to $H_p = 10$ and $H_u = 3$ respectively. Due to the difficulty faced in hand tuning the matrices Q and R, it was made more stable by reducing the perturbations ϵ_{max} and ϵ_{min} to 0. Finall, as we want to track the load angle of the servo, $C = [1, 0, 0, 0]^T$.

In the first experiment, I have run the standard MPC controller and the robust MPC controller with the nominal and actual models coinciding and their output and input plots versus time can be found in figure 2. In this scenario, $Q_k = 10000$ and $R_k = 0.1$ for the standard MPC and $Q_k = 50000$ and $R_k = 0.1$ for the robust MPC case. These parameters were hand-tuned by looking for the fastest and most steady tracking of the set point.

In the second experiment, I run the two controllers except with the fact that data is actually being generated by a model different from the nominal one, that is, from the non-linear model. The same type of comparison as done for the first experiment can be found in figure 3. In this scenario, $Q_k = 5000$ and $R_k = 0.01$ for the standard MPC and the same values were used for the robust MPC. The threshold for KL divergence, c was set to 0.1. These parameters were hand-tuned by visually assessing the quality of the output and input plots obtained. Similar to the previous experiment, I looked for the fastest and most steady tracking of the set point.

Figure 4 compares the root mean square error (RMSE) between the set point and the system output at every time step. This comparison is done for both experiments and both controllers. A zero error reference line has also been included to get a better idea about the magnitude of the RMSE. Table III compares the root mean square error across the entire time of simulation, the norm of the voltage input and the execution time of the controllers.

Now, let us first analyze the figures 2 and 3. Here, it can be seen that both the controllers in each of the experiments perform similarly. However, the system response of the output with the robust MPC is much faster in time compared to the standard MPC. In other words, the robust MPC reaches the set point value faster than the standard MPC. This is the case in both the experiments. This could imply that the robust controller is able to provide a better estimate of the state of the system than the standard Kalman filter with data obtained over fewer time steps.

In addition to these observations, a shaded region is depicted around the outputs from the two controllers in both experiments in figures 2 and 3. This shaded region has a total width of 4σ, with 2σ on either side of the mean estimated load angle state. This is used to provide a visualization of the confidence of the state estimator. σ here is the standard deviation of the load angle estimate and is obtained from matrix P_k as the square root of the element at the first row and column. It can be seen in figure 2 that the robust MPC has a large uncertainty in its estimate initially but also reduces rapidly with more time steps. This uncertainty reduces to a great extent after 10 seconds in the robust MPC. In the standard MPC however, the uncertainty is low from the start in this experiment. In figure 3, high uncertainty is seen once again here, but in both controllers. This uncertainty decreases rapidly with more time steps.

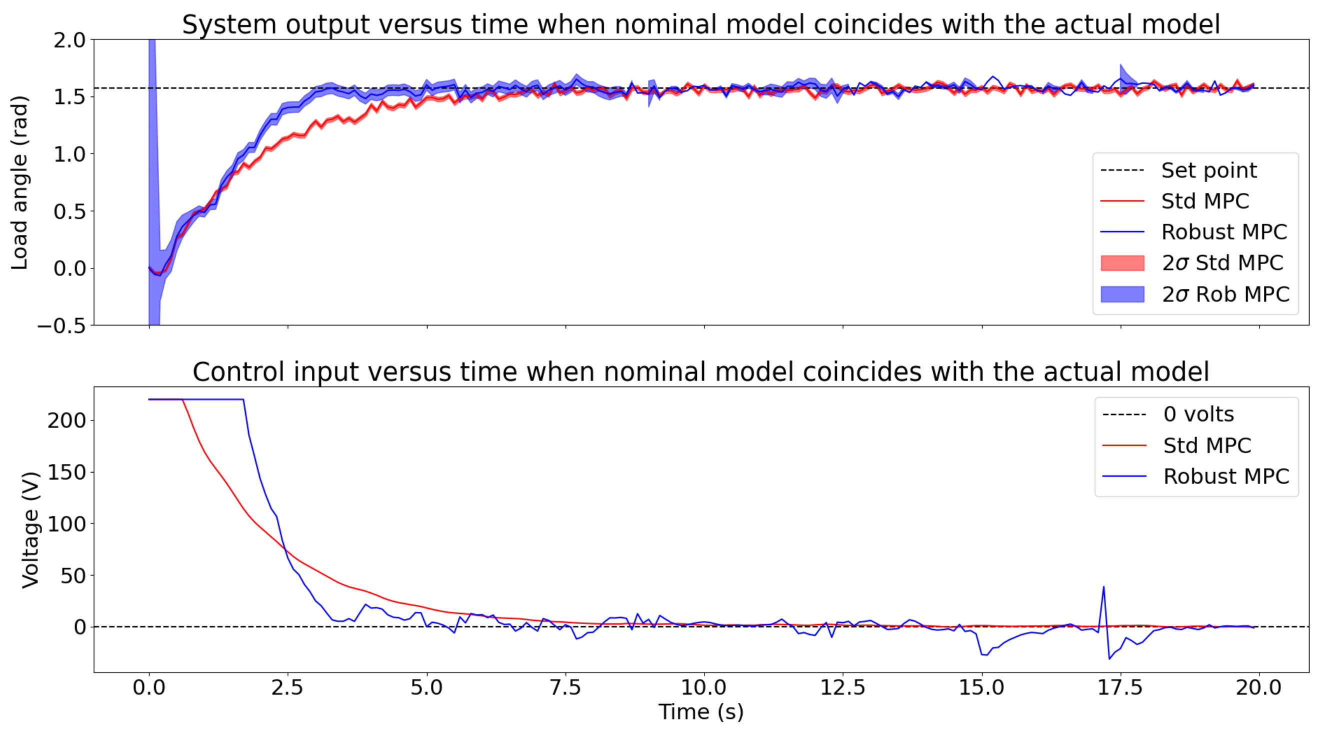


Fig. 2: Plots of the system output and the control input when the nominal model coincides with the actual model.

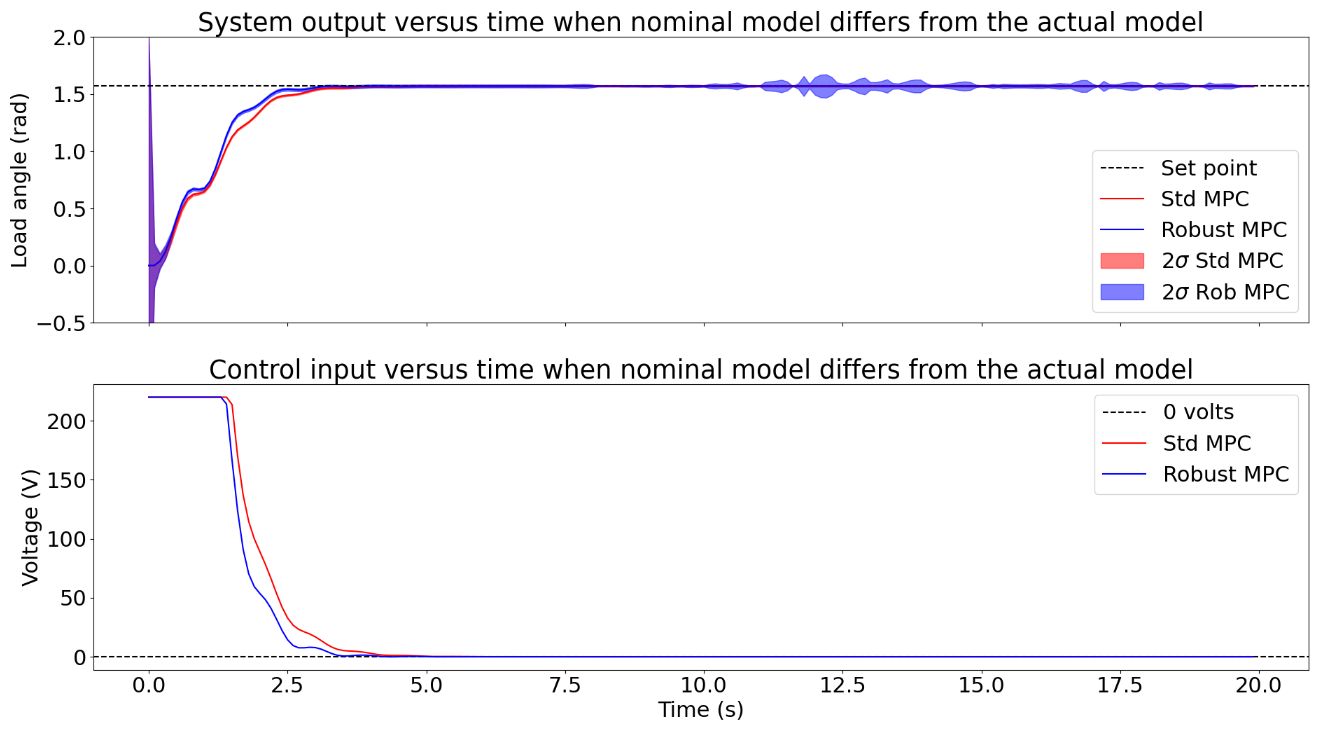
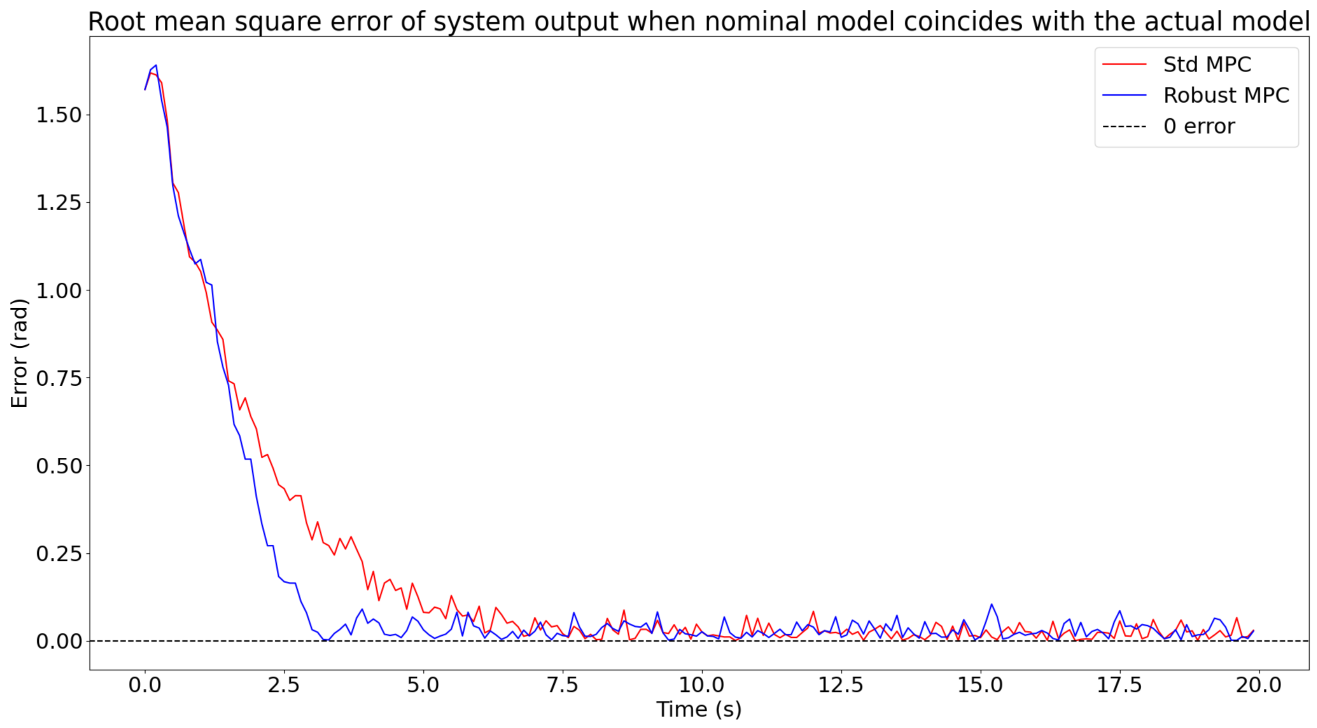
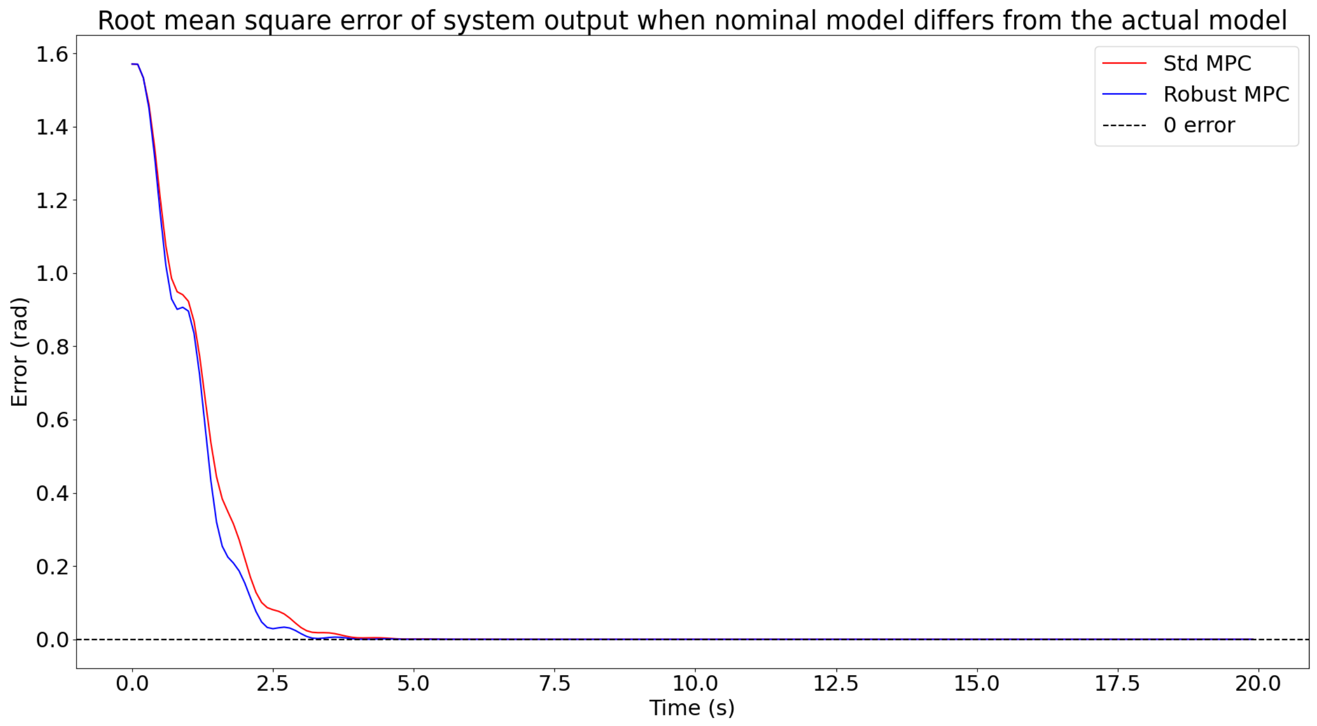


Fig. 3: Plots of the system output and the control input when the nominal model differs from the actual model.



(a) Nominal model coincides with the actual model.



(b) Nominal model differs from the actual model.

Fig. 4: Comparison of the root mean square error between the standard MPC and robust MPC for the two scenarios: when the nominal model coincides with the actual model and when the nominal model differs from the actual model.

TABLE III: Analysis of the root mean square error, norm of control input and execution time across all experiments.

Experiment	Controller	RMSE for tracking (rad)	Norm of control input (V)	Execution time (s)
Exp1: Nominal model coincides with the actual model	Standard MPC	0.3856	850.4994	0.000137
	Robust MPC	0.3631	1013.9673	0.001609
Exp2: Nominal model differs from the actual model	Standard MPC	0.3183	932.5745	0.000176
	Robust MPC	0.3074	890.1463	0.002250

However, the robust MPC in experiment 2 experiences some time intervals of lower confidence later in the simulation (between 10 - 20 seconds) and this is reflected as a wider shaded region in blue. The explanation for this could be attributed to numerical instability in the bisection algorithm as the value of τ_k determines the value of the covariance matrix $\tilde{P}$.

If we look at the input voltage with respect to time, we see that the input is much steeper in the robust MPC than the standard MPC. Moreover, the robust MPC saturates to the maximum voltage of 220V for a longer time period than the standard MPC in experiment 1. The input signal generated by the robust MPC is also much noisier than the standard MPC and can be explained by the fact that the robust MPC is constructed to account for model uncertainties.

The root mean square error plots in figure 4 and values in table III give us a better idea of the tracking ability of the controllers. The figure allows us to do temporal analysis on the root mean square error by seeing how it varies with time. It can be seen that in both experiments, the RMSE decreases as time increases, being close to 0 after about 10 seconds, which corresponds to its steady state. An interesting observation is that in both experiments, the robust MPC decreases more steeply to 0 compared to the standard MPC. Now, if we look at the table, we can see the RMSE across the entire simulation. It can be seen that the robust MPC in both experiments improves over the standard MPC by a slight amount.

The remaining two columns in table III give us an idea about the input voltage and the execution time of the controllers, which are more important when these are implemented in real-time systems. It can be seen that for experiment 1, more voltage is required over the entire time for the robust MPC case compared to the standard MPC case. However, the opposite happens in experiment 2. A lower norm on the voltage input over the entire time period is desirable as that would correspond to a lower current supply and would make the system more electrically efficient. Finally, if we analyze the execution times of the controllers, we can see that the robust MPC controllers takes much longer ($\sim$ 15 times longer) than the standard MPC. This is because of the root finding problem to determine τ and the inverse calculation to determine $\tilde{P}_k$ at every time step.

V. Conclusion

In this project, the robust Kalman filter was studied and implemented in an unconstrained MPC framework for a tracking problem with the servo-mechanism system. This filter was compared with its counter part in the standard MPC for the same application. It was observed that the robust MPC outperforms the standard MPC slightly but at the cost of higher execution times. To get a better idea about the advantages of this robust Kalman filter, future work could compare this with the standard MPC for many applications and model uncertainties. Moreover, the choice between the robust MPC and the standard MPC for a certain application can be made based on a number of factors such as the confidence in the nominal model in describing the system, amount of computation available, response time requirements etc.

References

[1] A. Zenere and M. Zorzi, "On the coupling of model predictive control and robust kalman filtering," *IET Control Theory & Applications*, vol. 12, no. 13, pp. 1873–1881, 2018.

[2] A. BEMPORAD and E. MOSCA, "Fulfilling hard constraints in uncertain linear systems by reference managing," *Automatica*, vol. 34, no. 4, pp. 451–461, 1998.

[3] B. C. Levy and R. Nikoukhah, "Robust state space filtering under incremental model perturbations subject to a relative entropy tolerance," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 682–695, 2013.

[4] S. A. Kassam and T. L. Lim, "Robust wiener filters," *Journal of the Franklin Institute*, vol. 304, no. 4, pp. 171–185, 1977.

[5] S. Kassam and H. Poor, "Robust techniques for signal processing: A survey," *Proceedings of the IEEE*, vol. 73, no. 3, pp. 433–481, 1985.