Master's Thesis
석사 학위논문

Modeling and Precision Stop of Metropolitan Train with Receding Horizon Control

Sungmin Aum(엄 성 민 嚴 珹 珉)

Department of Information
and Communication
Engineering
정보통신융합공학전공

DGIST

2015

Master's Thesis
석사 학위논문

Modeling and Precision Stop of Metropolitan Train with Receding Horizon Control

Sungmin Aum(엄 성 민 嚴 珹 珉)

Department of Information
and Communication
Engineering
정보통신융합공학전공

DGIST

2015

MS/IC
201322023

엄 성 민. Sungmin Aum. Modeling and Precision Stop of Metropolitan Train with Receding Horizon Control. Department of Information and Communication Engineering. 2015. 85p. Advisor Prof. Eun, Yongsoon. Co-Advisor Prof. Hongsoo Choi

ABSTRACT

Modeling of new type of metropolitan train, which is to be introduced on Korean railways in near future, and the control strategy for precision stop is tested in simulation. Modeling is carried out based on the raw test data and previously measured specifications provided by the Korean Railroad Research Institute. Control strategy is divided to two stages. Feed-forward and PI controller was implemented for initial stage. The controller is then switched to a Linear Receding Horizon Controller closed to the train stop, when velocity is low enough to disregard the actuator and plant nonlinearities. New reference is generated online for Linear Receding Horizon Controller interval. Simulations under the reasonable amount of measurement noise and model mismatch show that the proposed controller satisfies precision stop specification of ± 0.1m.

Keywords: metro train, train precision stop, receding horizon control, model predictive control

Contents

List of Figures

1 Introduction

1.1 Motivation

Metro train is heavily used by citizen all around the world. Seoul metro alone transported more than 1.87 billion people in 2013 [1]. Safe and timely function of metro train is therefore a significant matter.

The Korean government seeks to introduce a new type of train to its metro train system[2]. Among many improvements, one of the most significant differences is its widened door. All of the previous models have a door width of 1.3m, but the new model's door width is 1.8m. In theory, the widened door would allow three lines of passengers to move through simultaneously. The current door is designed for two lines of passengers to move though simultaneously.

Most of the metro stations have Platform Screen Door in Korea, and soon all stations will have them installed. The width of each PSD is fixed at 2.0m. This means that when the new train is introduced, the maximum tolerable stop error margin decreases from current ± 0.35m to ± 0.1m. Therefore, more sophisticated control strategy is in need in order to meet the new control specification. When successful, the introduction of new train could possibly bring significant economic benefits to the nation with increased public transportation throughput.

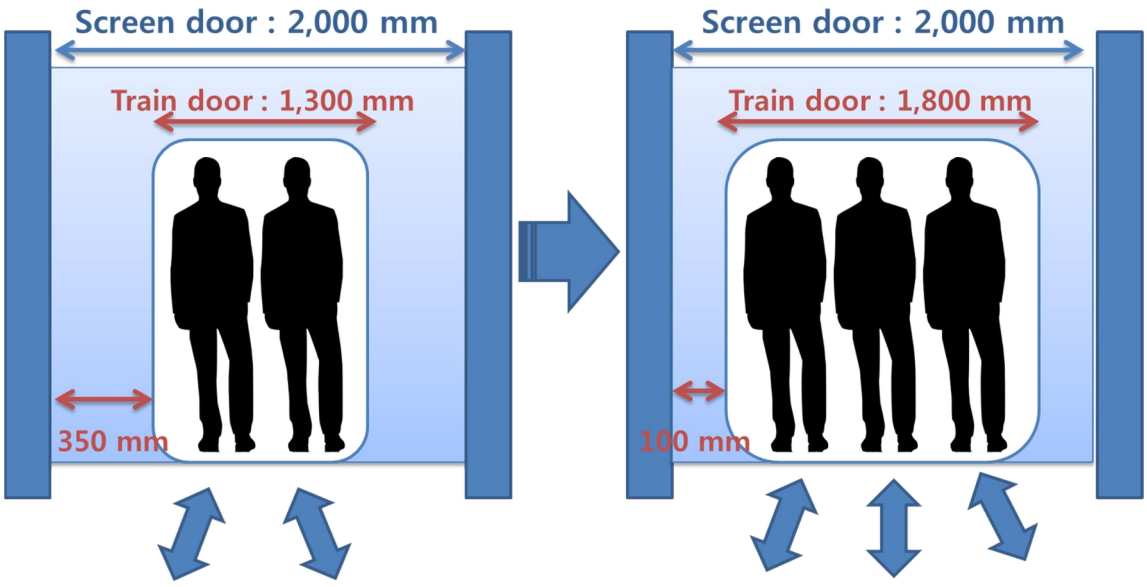


Figure 1 Effect of extended train doors

In order to successfully implement precision control strategy, appropriate modeling must take place. When important factors are not considered, unexpected behavior may arise in control system. But if the model in consideration is too complex, analysis on such control system would be difficult, and control implementation itself maybe too costly or even impossible due to computational limitations. Therefore it is necessary to try to incorporate as many significant factors as possible while trying to simplify the model with implementation in mind.

1.2 Contribution

The main contributions of this thesis are:

1. More adequate modeling of the new train for control purpose. In specific, the consideration of deadtime dynamics of actuators.
2. Exploring the possibility of applying Receding Horizon Control for metro train precision stop purpose in practical perspective.

1.3 Methodology

The model in consideration includes following:

- 6-car train, each car linked with connectors.
- Passenger mass.
- Blending of regenerative and air brakes.
- Actuator deadtime.
- Different transient responses between rising and falling interval, i.e., when force is applied and when force is released. This is due to deadtime.
- Mismatch in ω_n value of the actuators' transient responses in standard 2nd order transfer function.
- Mismatch in car and passenger mass.
- Mismatch in deadtime.

Parameters used for modeling are provided by KRRI in a form of graph from experiments or measured values.

The control strategy implemented is as follows:

1. 2DOF controller(Feedforward+PI) phase
2. Receding Horizon Controller phase(speed<2.5m/s, after passing PSM3)

1.4 Summary

The overview of the rest of this thesis is as follows. Chapter II will provide brief introduction to metropolitan train system, so unfamiliar readers would have understanding of metro train stop process. Chapter III will provide the model of new metro train of interest. Chapter IV will demonstrate the proposed control strategy suitable for the model given in Chapter III. Chapter V will then describe how the simulation is implemented, and the results will be shown in Chapter VI. Chapter VII will have concluding remarks.

2 Background Information

2.1 Previous Work

Train control, being over a hundred year old topic [26], may be reaching a state of maturity in present time. Indeed there have been many previous work related to train control. For precision stop purpose, appropriate modeling is essential. If a model is too coarse, any controller would have difficulties in satisfying control specifications, and for some advance control technique may not even be feasible. If a model is too detailed, analysis of the effect of controller on such model may be limited, and the computational cost could be too expensive for actual implementation. Therefore, a model which captures the essence of train dynamics yet simple enough for actual implementation is crucial for precision stop.

Among many nonlinearities which exist in train, brake blending, delayed reaction of actuator and varying degree of actuator saturation related to the velocity of train are one of the most important factors to consider when modeling train for control purpose. Leaving any one of the above factors out may result in ineffective modeling. However, to the best of authors knowledge, no previous work consider all of the above matters together in terms of suitable mathematical representation fit for control purpose. Table 1 summarizes the incompleteness of previous work.

Modeling work in [3,4,6,11~16,23] are not exclusively done for control purpose, so although some have equations present within, they are generally not satisfactory to use in controller design directly. Trains have both utilize mechanical (disk/air) brakes and electrical (resistive/regenerative) brakes, therefore blending needs to be considered. [5~10,13~16,24] do not consider brake blending. Brakes and traction motor used in train display varying degree of saturation respect to speed. [7~10,13,15,16] do not mention this actuator nonlinearity related to the velocity of the vehicle. The input command of train do not get an immediate response from actuators. [3,5,7,8,11,13~16] do not consider time delay. By conducting close examinations on experimental data, it may be reasonable to model the actuation delay as deadtime and asymmetric transient response [24] rather than pure time delay coupled with symmetric transient response in [6,9,10]. More elaboration on deadtime phenomenon is discussed in 3.2.3.

2.2 Automatic Train Control (ATC)

Automatic Train Control, or ATC, works as a safety mechanism which oversees the entire train system. ATC monitors condition of the rails and position of each train. ATC sets maximum velocity for given interval, taking consideration of various information it gathered. The set maximum velocity is then delivered to Automatic Train Operation.

2.3 Automatic Train Operation (ATO)

The most significant part of metropolitan rail system is arguably Automatic Train Operator, or ATO. ATO is in charge of providing hardware environment for train controller and communication and recording of relevant data. ATO receives position data from PSM, and deliver them to the train controllers.

2.4 Precision Stop Marker (PSM)

There are four PSMs used in Korean metro system. Each PSM has its own ID and are positioned as below [Figure 2]. If there are two separate entries to one station, PSM1 is located before the rail gets separated and are shared by both rails. Operations carried out by passing each PSM is as below

PSM1: Reset remaining distance counter to 541m. Load velocity profile depends on the current speed and stopping station. Counter continues to calculate position afterward by encoders from wheel.

PSM2: Proceed with control. When slowing down, only use regenerative brake

PSM3: Stop using regenerative brake and use air brake.

PSM4: Reset remaining distance to 3.5m. Only air brake is used.

Station: Stop at a designated position. Position error tolerance is ±0.35

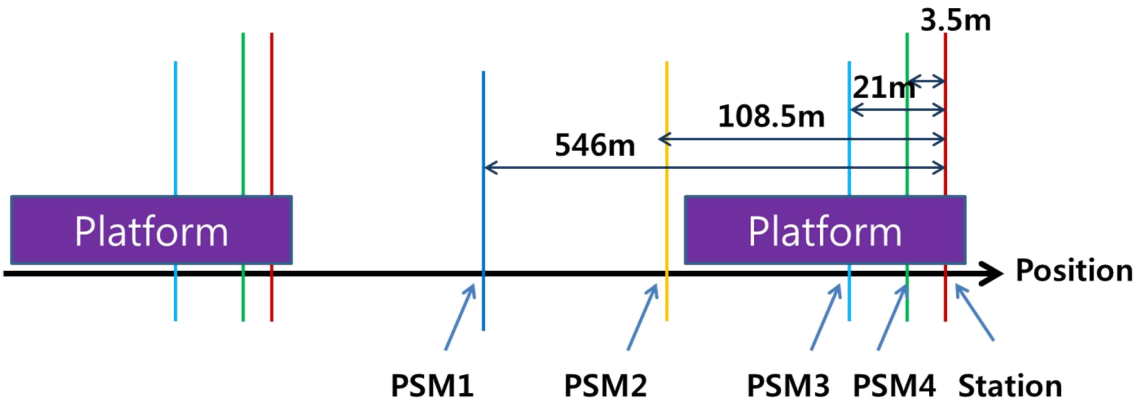


Figure 2 PSM Locations

Here is a big picture of metro train control system: A metro train follows premade velocity reference. The maximum speed is set at 80km/h by ATC. ATO provides appropriate velocity reference and measurements from PSMs. Velocity is provided by encoder on the wheel. Brakes and traction motor has their own dynamics and are controlled by separate controllers. In all stations, the train must stop within 0.35m of predetermined, fixed position. Train controller follows velocity reference while operating under above constraints.

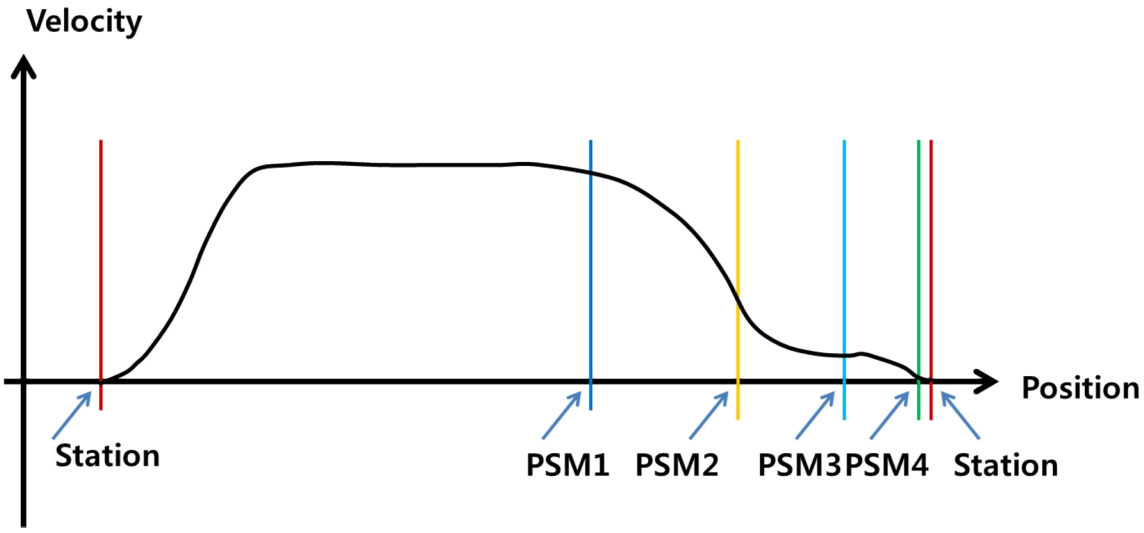


Figure 3 An Example of Velocity Reference

2.5 Car types

There are two different kinds of train car. One is 'Motor car' or 'M-car' and other is 'Trail car' or 'T-car'. The M-car has motor, regenerative brake, and air brake. The T-car does not have motor and regenerative brake but has disk brakes which are generally stronger than air brakes in M-car.

The new metropolitan train model is consisted of all M-cars, which makes it possible to control each car with separate controllers. This property is exploited for RHC design. Details are explained in Chapter IV.

2.6 Train formation

Previous metro trains used on Korea are consisted of lead vehicle and tail vehicle. [Figure 4-a] An M-car plays a role of the lead vehicle and T-car plays a role of tail vehicle within a pair. Each pair receives the controller command as one unit.

The new model, however, is consisted entirely of M-cars[Figure 4-b] The paring is no longer necessary because every vehicles are equipped with both the traction motor and brakes.

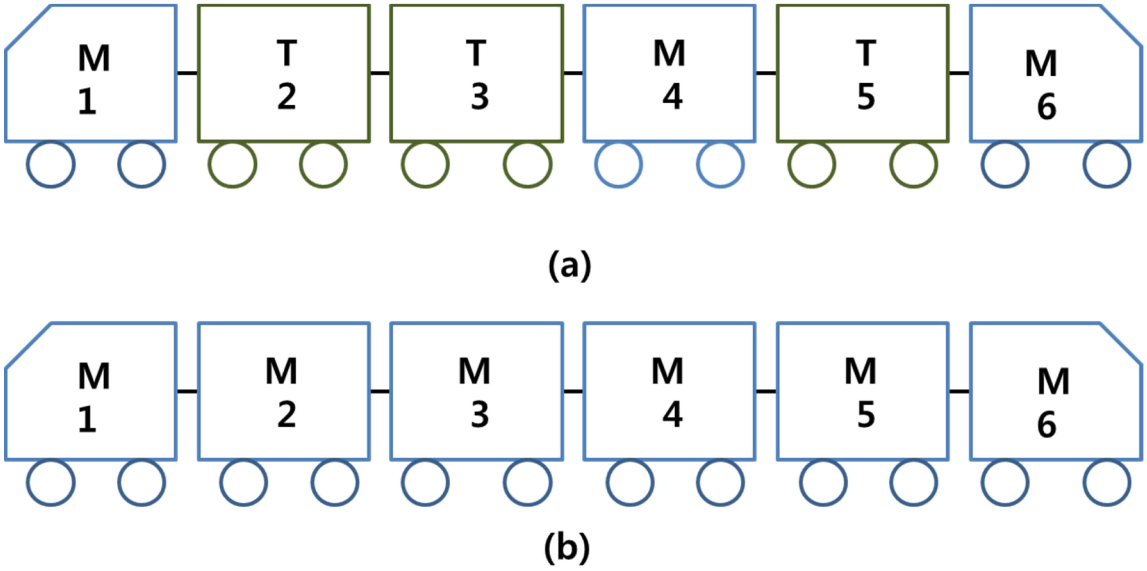


Figure 4 Two kinds of train formation model. (a) MT type, (b) MM type

2.7 Brakes

There are two types of brakes used in metro train in consideration. One is air brake, and the other is regenerative brake.

An air brake or, more formally, a compressed air brake system, is a type of friction brake for vehicles in which compressed air pressing on a piston is used to apply the pressure to the brake pad needed to stop the vehicle. An air brake is consisted of piston, brake pad, and air compression compartment. Compressed air press on piston to apply pressure on the brake pad to deliver the stopping force.

While air brake generates stopping force by applying friction, regenerative brake does not. One may think of regenerative brake as a generator. Part of the kinetic energy of the train is converted to electrical energy by reversing the function of traction motor. The generated electrical energy is either delivered to the passing train or elsewhere. Since none of its part is subject to wearing, regenerative brake has longer lifespan compare to other friction brakes.

3 Modeling

3.1 Train model

Our train model considers six M-cars connected with non-rigid couplers. The model equations are given below:

$$\dot{p}_i = v_i \tag{3.1}$$

$$m_i \dot{v}_i = k_{i-1}(p_{i-1} - p_i) + k_i(p_{i+1} - p_i) + c_{i-1}(v_{i-1} - v_i) + c_i(v_{i+1} - v_i) + F_i(v_i) \tag{3.2}$$

$$F_i(v_i, \mathrm{t}) = F_i^{tr}(v_i, \mathrm{t}) + F_i^{rb}(v_i, \mathrm{t}) + F_i^{ab}(v_i, \mathrm{t}) + F_i^{\mathrm{d}}(v_i, \mathrm{t}), \tag{3.3}$$

$$F_i^{tr}(v_i, \mathrm{t}) = \mathrm{C_i^{tr}} \zeta_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}) \tag{3.4}$$

$$\dot{\zeta}_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}) = \mathrm{A_i^{tr}}\left(\alpha_\mathrm{i}^{\mathrm{tr}}(\mathrm{t})\right)\zeta_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}) + \mathrm{B_i^{tr}}\left(\alpha_\mathrm{i}^{\mathrm{tr}}(\mathrm{t})\right)\Delta_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}) \tag{3.5}$$

$$\Delta_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}) \begin{cases} 0, \text{if } F_i^{tr'}(\tau) = 0 \; for \; some \; \tau \in [t - t_{i,d}^{tr}, t] \\ F_i^{tr'}(\tau) \; otherwise \end{cases} \tag{3.6}$$

$$\alpha_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}) \begin{cases} \mathrm{r}, \; \Delta_\mathrm{i}^{\mathrm{tr}} > F_i^{tr} \\ \mathrm{f}, \; \Delta_\mathrm{i}^{\mathrm{tr}} < F_i^{tr} \\ \alpha_\mathrm{i}^{\mathrm{tr}}(\mathrm{t}^-), \Delta_\mathrm{i}^{\mathrm{tr}} = F_i^{tr} \end{cases} \tag{3.7}$$

$$\mathrm{A_i^{tr}(r)} = \mathrm{A_i^{tr_r}} \tag{3.8}$$

$$\mathrm{B_i^{tr}(r)} = \mathrm{B_i^{tr_r}} \tag{3.9}$$

$$\mathrm{A_i^{tr}(f)} = \mathrm{A_i^{tr_f}} \tag{3.10}$$

$$\mathrm{B_i^{tr}(f)} = \mathrm{B_i^{tr_f}} \tag{3.11}$$

$$F_i^{rb}(v_i, \mathrm{t}) = \mathrm{C_i^{rb}}\zeta_\mathrm{i}^{\mathrm{rb}}(\mathrm{t}) \tag{3.12}$$

$$\dot{\zeta}_\mathrm{i}^{\mathrm{rb}}(\mathrm{t}) = \mathrm{A_i^{rb}}\left(\alpha_\mathrm{i}^{\mathrm{rb}}(\mathrm{t})\right)\zeta_\mathrm{i}^{\mathrm{rb}}(\mathrm{t}) + \mathrm{B_i^{rb}}\left(\alpha_\mathrm{i}^{\mathrm{rb}}(\mathrm{t})\right)\Delta_\mathrm{i}^{\mathrm{rb}}(\mathrm{t}) \tag{3.13}$$

$$\Delta_\mathrm{i}^{\mathrm{rb}}(\mathrm{t})\begin{cases} 0, \text{if } F_i^{rb'}(\tau) = 0 \; for \; some \; \tau \in [t - t_{i,d}^{rb}, t] \\ F_i^{rb'}(\tau) \; otherwise \end{cases} \tag{3.14}$$

$$\alpha_\mathrm{i}^{\mathrm{rb}}(\mathrm{t})\begin{cases} \mathrm{r}, \; \Delta_\mathrm{i}^{\mathrm{rb}} > F_i^{rb} \\ \mathrm{f}, \; \Delta_\mathrm{i}^{\mathrm{rb}} < F_i^{rb} \\ \alpha_\mathrm{i}^{\mathrm{rb}}(\mathrm{t}^-), \Delta_\mathrm{i}^{\mathrm{rb}} = F_i^{rb} \end{cases} \tag{3.15}$$

$$\mathrm{A_i^{rb}}(\mathrm{r}) = \mathrm{A_i^{rb_r}} \tag{3.16}$$

$$\mathrm{B_i^{rb}}(\mathrm{r}) = \mathrm{B_i^{rb_r}} \tag{3.17}$$

$$\mathrm{A_i^{rb}}(\mathrm{f}) = \mathrm{A_i^{rb_f}} \tag{3.18}$$

$$\mathrm{B_i^{rb}}(\mathrm{f}) = \mathrm{B_i^{rb_f}} \tag{3.19}$$

$$F_i^{ab}(v_i, \mathrm{t}) = \mathrm{C_i^{ab}}\zeta_\mathrm{i}^{\mathrm{ab}}(\mathrm{t}) \tag{3.20}$$

$$\dot{\zeta}_\mathrm{i}^{\mathrm{ab}}(\mathrm{t}) = \mathrm{A_i^{ab}}\left(\alpha_\mathrm{i}^{\mathrm{ab}}(\mathrm{t})\right)\zeta_\mathrm{i}^{\mathrm{ab}}(\mathrm{t}) + \mathrm{B_i^{ab}}\left(\alpha_\mathrm{i}^{\mathrm{ab}}(\mathrm{t})\right)\Delta_\mathrm{i}^{\mathrm{ab}}(\mathrm{t}) \tag{3.21}$$

$$\Delta_\mathrm{i}^{\mathrm{ab}}(\mathrm{t})\begin{cases} 0, \text{if } F_i^{ab'}(\tau) = 0 \; for \; some \; \tau \in [t - t_{i,d}^{ab}, t] \\ F_i^{ab'}(\tau) \; otherwise \end{cases} \tag{3.22}$$

$$\alpha_\mathrm{i}^{\mathrm{ab}}(\mathrm{t})\begin{cases} \mathrm{r}, \; \Delta_\mathrm{i}^{\mathrm{ab}} > F_i^{ab} \\ \mathrm{f}, \; \Delta_\mathrm{i}^{\mathrm{ab}} < F_i^{ab} \\ \alpha_\mathrm{i}^{\mathrm{ab}}(\mathrm{t}^-), \Delta_\mathrm{i}^{\mathrm{ab}} = F_i^{ab} \end{cases} \tag{3.23}$$

$$\mathrm{A}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r}) = \mathrm{A}_{\mathrm{i}}^{\mathrm{ab_r}} \tag{3.24}$$

$$\mathrm{B}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r}) = \mathrm{B}_{\mathrm{i}}^{\mathrm{ab_r}} \tag{3.25}$$

$$\mathrm{A}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{f}) = \mathrm{A}_{\mathrm{i}}^{\mathrm{ab_f}} \tag{3.26}$$

$$\mathrm{B}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{f}) = \mathrm{B}_{\mathrm{i}}^{\mathrm{ab_f}} \tag{3.27}$$

i=1…6,

where $\mathrm{m_i}, p_i, v_i$ are i^{th} vehicle's mass, position, and velocity. k_i and C_i are spring constant and damping constant of couplers between i^{th} vehicle and $i+1^{th}$ vehicle. F_i is the total force exerted upon the i^{th} vehicle. F_i^{tr},F_i^{rb}, and F_i^{ab} are traction, regenerative brake, and air brake force. $F_i^{tr'}$,$F_i^{rb'}$, and $F_i^{ab'}$ are desired traction, regenerative brake, and air brake force command given by controller. F_i^{d} is the running resistance. $\Delta_{\mathrm{i}}^{\mathrm{tr}}(\mathrm{t})$, $\Delta_{\mathrm{i}}^{\mathrm{rb}}(\mathrm{t})$, and $\Delta_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{t})$ are functions introduced to describe the deadtime phenomenon. $\alpha_{\mathrm{i}}^{\mathrm{tr}}(\mathrm{t})$, $\alpha_{\mathrm{i}}^{\mathrm{rb}}(\mathrm{t})$, $\alpha_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{t})$, together with $\mathrm{A}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r})$, $\mathrm{B}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r})$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r})$, $\mathrm{B}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r})$,. $\mathrm{A}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r})$, and $\mathrm{B}_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{r})$ are functions that describe asymmetric transient response. $t_{i,d}^{tr}$, $t_{i,d}^{rb}$, and $t_{i,d}^{ab}$ are the amount of deadtime of respective actuators. $\zeta_{\mathrm{i}}^{\mathrm{tr}}(\mathrm{t})$, $\zeta_{\mathrm{i}}^{\mathrm{rb}}(\mathrm{t})$, $\zeta_{\mathrm{i}}^{\mathrm{ab}}(\mathrm{t})$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{tr_r}}$, $\mathrm{B}_{\mathrm{i}}^{\mathrm{tr_r}}$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{tr_f}}$, $\mathrm{B}_{\mathrm{i}}^{\mathrm{tr_f}}$, $\mathrm{C}_{\mathrm{i}}^{\mathrm{tr}}$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{rb_r}}$, $\mathrm{B}_{\mathrm{i}}^{\mathrm{rb_r}}$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{rb_f}}$, $\mathrm{B}_{\mathrm{i}}^{\mathrm{rb_f}}$, $\mathrm{C}_{\mathrm{i}}^{\mathrm{rb}}$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{ab_r}}$,$\mathrm{B}_{\mathrm{i}}^{\mathrm{ab_r}}$, $\mathrm{A}_{\mathrm{i}}^{\mathrm{ab_f}}$,$\mathrm{B}_{\mathrm{i}}^{\mathrm{ab_f}}$, and $\mathrm{C}_{\mathrm{i}}^{\mathrm{ab}}$ are the state variables and matrices corresponding to the transient responses of the respective actuators when the absolute value of the force is rising and falling. The asymmetric transient response was the key finding of [24] along with the deadtime phenomenon of the train actuators.

3.2 Actuator characteristics

3.2.1 Transient response

The transient responses of the actuators were driven from the raw experimental datasheet provided by KRRI. By fitting the raw data with second order transfer functions, the transient responses of traction motor, regenerative brake, and air brake were approximated.

3.2.2 Nonlinearity depending on velocity

The actuators exhibit nonlinearity depending on velocity of the train. In specific, the input saturation levels depend on the velocity. The traction motor, regenerative brake, and air brake all show decrease in maximum performance under high velocity. Regenerative brake shows decrease in maximum performance when velocity is lower

than about 2.5m/s, but traction motor and air brake shows practically linear characteristics when velocity is below certain amount [Figure 5~7].

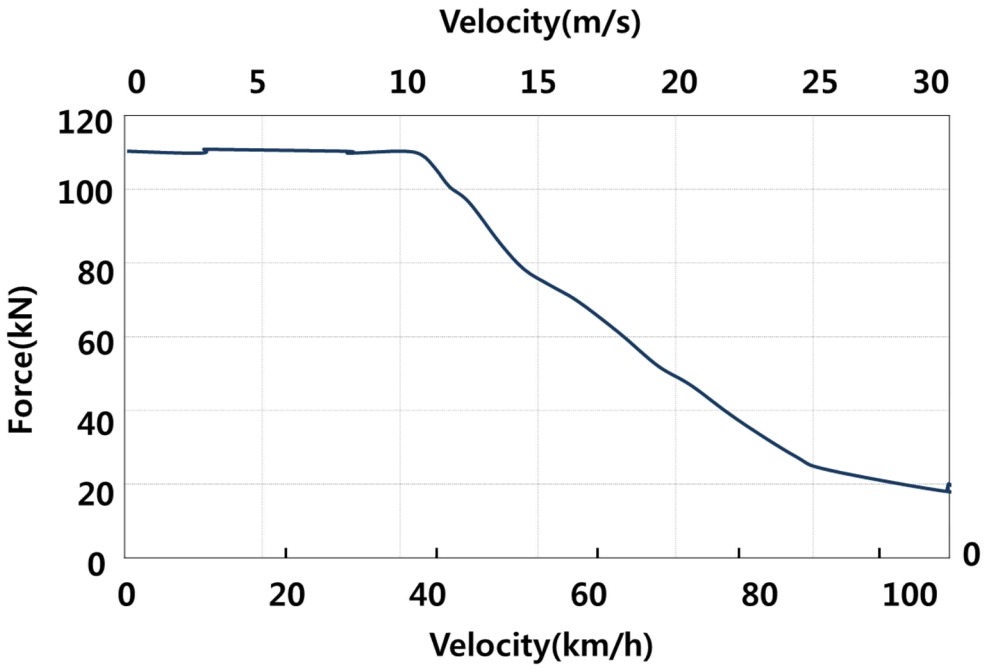


Figure 5 Traction motor force saturation respect to velocity

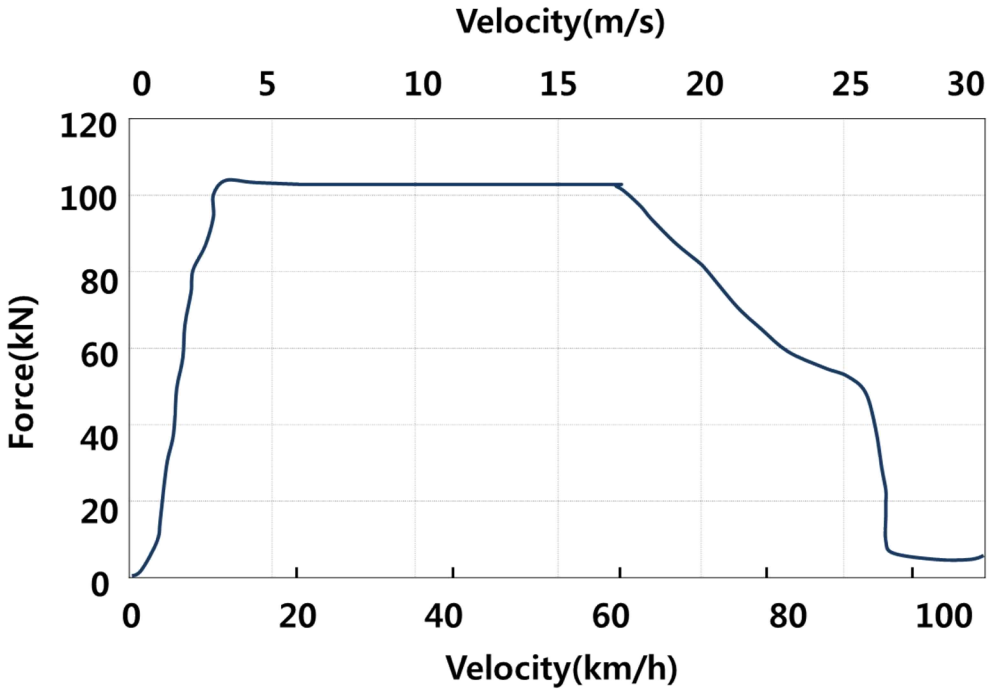


Figure 6 Regenerative brake force saturation respect to velocity

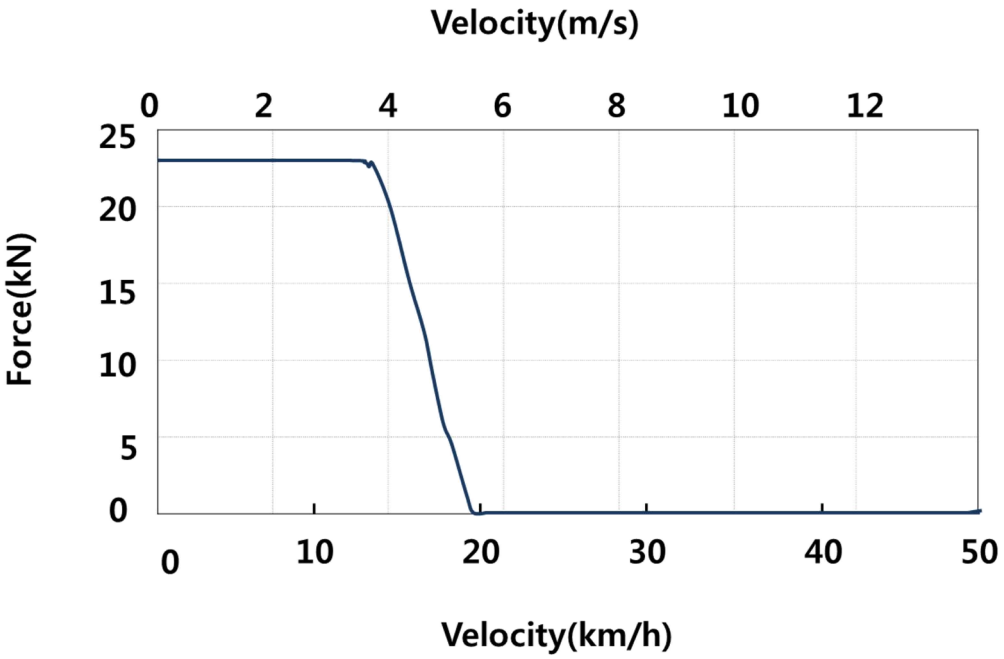


Figure 7 Air brake force saturation respect to velocity

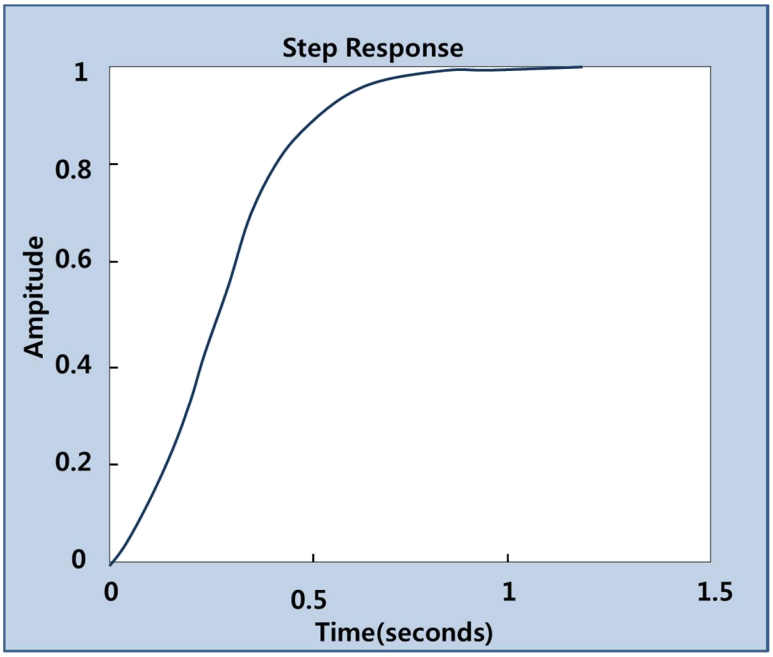


Figure 8 Approximated transient response of traction motor

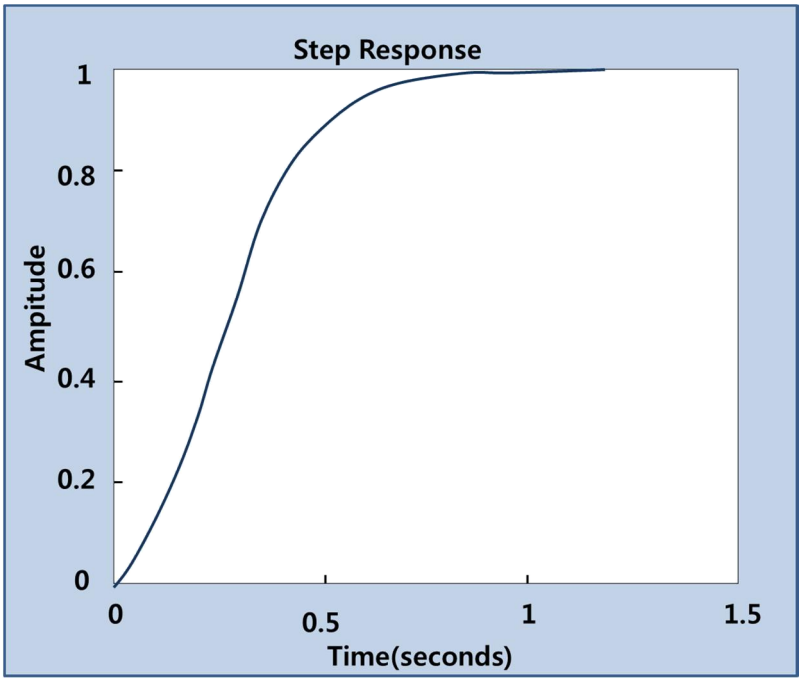


Figure 9 Approximated transient response of regenerative brake

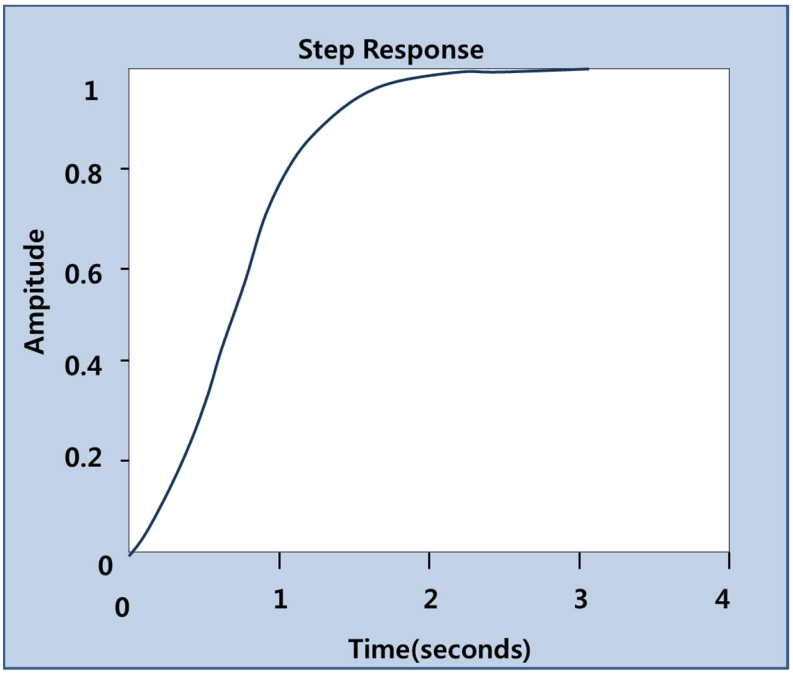


Figure 10 Approximated transient response of air brake

3.2.3 Deadtime

Many existing work [4,6,9,10,12,23] recognize the time delay of train actuators. It is common practice to model the actuation delay as a combination of pure delay and symmetric transient response [6,9,10]. Here the symmetric transient response means that the transient response of the actuator is same whether the force is being applied or not. However, the experimental data from [23] exhibits two crucial difference from the combination of pure delay and symmetric transient response. First, the brake force shows no response for significant amount of time when the PWM signal is initially applied, but displays no notable amount of delay when PWM signal is changed so that the desired brake force is zero. Second, it is clear from the graph that the transient response of brake force is much faster when the PWM signal ceases to require non-zero amount of force. [23]. This phenomenon was first pointed out by [24].

The comparison between traditional model (pure delay + symmetric transient response) and model used in this thesis (deadtime + asymmetric transient response) is shown in Figure 11.

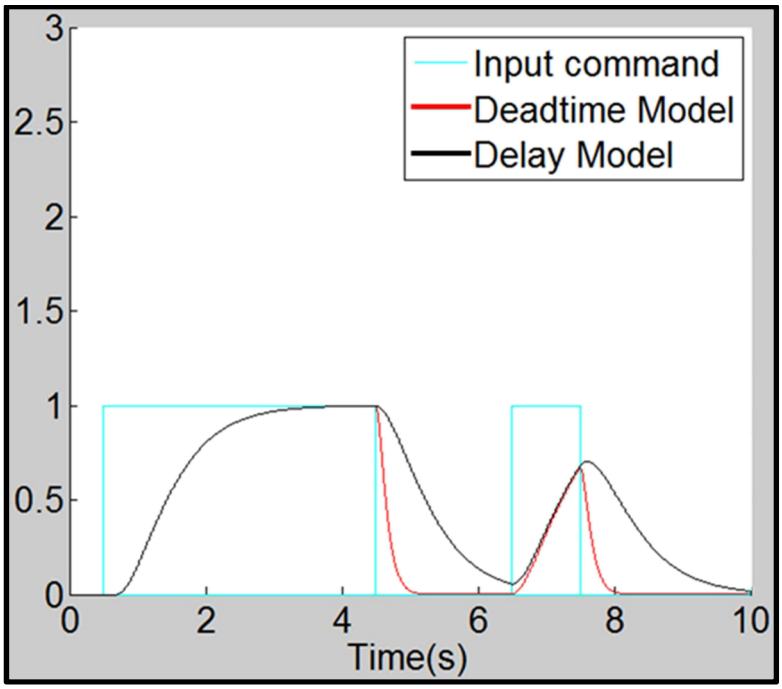


Figure 11 Comparison between traditional train actuator model and the model used in this thesis

The implementation of deadtime phenomenon under discrete time domain is given in below pseudo code:

```
SET current iteration time, air brake deadtime count, regenerative brake deadtime count,
    traction deadtime count =0
    FOR each car in a train
        READ current iteration time, air brake deadtime count,
       regenerative brake deadtime count, traction deadtime count,
        IF current iteration time < 3
            increase air brake deadtime count by 1
            increase regenerative brake deadtime count by 1
            increase traction deadtime count by 1
    ELSE
    READ desired and actual air brake force, regenerative brake force, traction force
    READ state variables of air brake force, regenerative brake force, traction force
        {for air brake}
        IF previous desired air brake force is 0 and air brake force before
            the previously desired air brake force is not 0
           update air brake state variable with falling dynamics model
           SET air brake deadtime count = 0
        ELSEIF previous desired air brake force is not 0 and air brake force before
                    the previously desired air brake force is 0, but air brake deadtime
                    count is not full update air brake state variable with rising dynamics
                    model increase air brake deadtime count by 1
    ELSEIF previous desired air brake force is not 0 and air brake force before the
              previously desired air brake force is also not 0
              update air brake state variable with rising dynamics model
              SET air brake deadtime count = 0
    ELSE update air brake state variable with falling dynamics model
            SET air brake deadtime count = 0
    ENDIF {end of air brake deadtime}
        {for regenerative brake}
    IF previous desired regenerative brake force is 0 and regenerative brake force
        before the previously desired regenerative brake force is not 0
        update regenerative brake state variable with falling dynamics model
       SET regenerative brake deadtime count = 0
    ELSEIF previous desired regenerative brake force is not 0 and regenerative brake
              force before the previously desired regenerative brake force is 0,
              but regenerative brake deadtime count is not full
              update regenerative brake state variable with rising dynamics model
              increase regenerative brake deadtime count by 1
```

```
ELSEIF previous desired regenerative brake force is not 0 and regenerative brake
        force before the previously desired regenerative brake force is also not 0
        update regenerative brake state variable with rising dynamics model
        SET regenerative brake deadtime count = 0
ELSE update regenerative brake state variable with falling dynamics model
     SET regenerative brake deadtime count = 0
ENDIF {end of regenerative brake deadtime}
  {for traction}
IF previous desired traction force is 0 and traction force before the previously
   desired traction force is not 0
   update traction state variable with falling dynamics model
    SET traction deadtime count = 0
ELSEIF previous desired traction force is not 0 and traction force before the
        previously desired traction force is 0, but traction deadtime count is not
        full
        update traction state variable with rising dynamics model
        increase traction deadtime count by 1
ELSEIF previous desired traction force is not 0 and traction force before the
        previously desired traction force is also not 0
        update traction state variable with rising dynamics model
        SET traction deadtime count = 0
ELSE update traction state variable with falling dynamics model
      SET traction deadtime count = 0
ENDIF {end of traction deadtime}
ENDFOR
```

3.3 Resistance

Running resistance is the combined resistance from track and air resistance. Equations derived from experiments are usually preferred[19]. Here, a second order polynomial function of velocity is used to approximate the running resistance of the lead car. Assuming the rest of the cars will not experience as significant amount of air resistance, a first order function is used for the rest. The approximation is carried out based on the experimental data provided by the KRRI.

$$\mathrm{F}_1^d(v_1) = m_1 \times (0.022v^2 + 0.036|v| + 0.961) \times 10^{-3} \quad (3.28)$$

$$\mathrm{F}_\mathrm{i}^d(v_i) = m_i \times (0.036|v| + 0.961) \times 10^{-3},\ i = 2, 3, 4, 5, 6 \quad (3.29)$$

Figure 12 shows the resistance force with respect to velocity.

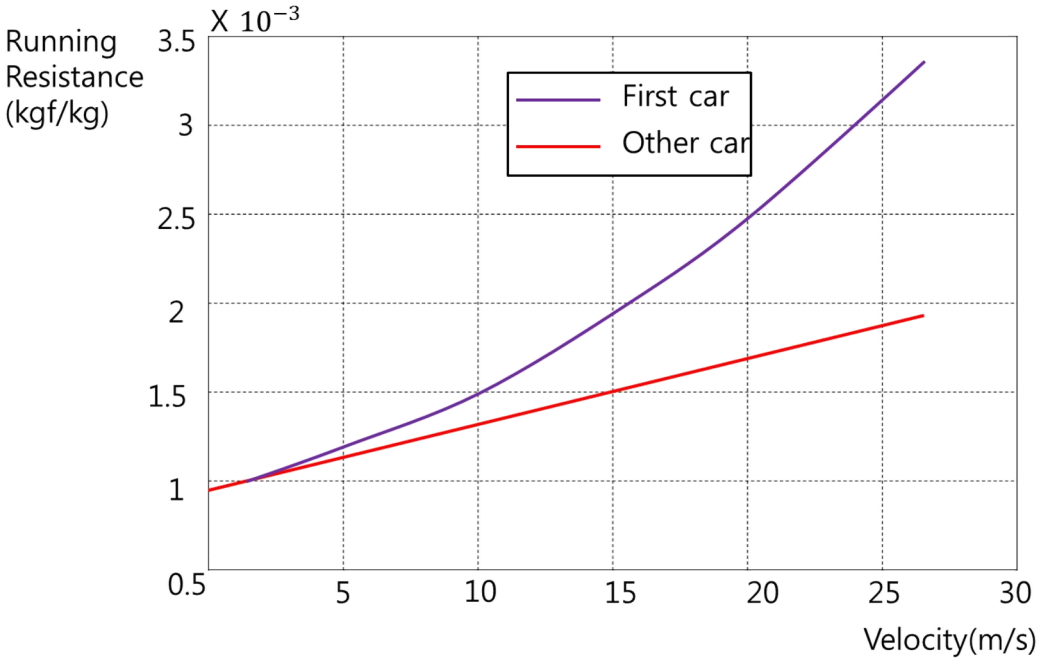


Figure 12 Running resistance

3.4 Brake blending

Two different kinds of brakes are utilized during train control. One is regenerative brake, and the other one is air brake. The brakes have different transient response and different constraints, so they must be treated separately, and appropriate measures to mix the braking force executed by each brakes under different operating circumstances have to exist.

The brake blending implemented in the simulation in this thesis is as follows:

Interval with velocity>2.5m/s:

- If requested braking force is less than that of maximum regenerative brake force on particular velocity at the time of request, only use regenerative brake force.
- If requested braking force is greater than that of maximum regenerative brake force on particular velocity at the time of request, rest of the requested force is applied by air brake.
- If requested braking force is equal to maximum regenerative brake force, only use regenerative brake.

Interval with velocity≤2.5m/s:

- Only use air brake.

Note that when the velocity of the vehicle is lower than 2.5m/s[Figure 6], the regenerative brake shows drastic decrease of its force. Furthermore, the original graph given by KRRI did not clearly show the regenerative brake force below 0.33m/s. On the other hand, the air brake force remains rather constant for such low velocity. Therefore, the air brakes are exclusively used as a control input when the velocity is low.

4 Controller Design

4.1 Control Strategy

The proposed control strategy for the precision stop of the new metropolitan train is two staged as shown below:

1. When a train leaves from a previous station, start with a 2DOF controller (Feedforward+PI)
2. When velocity is lowered to 2.5 m/s after passing PSM3, switch to Linear Receding Horizon Controller to finally bring the train to stop at the designated position.

Because the focus of this thesis is on the precision stop of the train, the first stage could be an arbitrary control strategy, which could also possibly be replaced by just a PI controller as it is currently deployed[2].

The result of above control strategy under nominal condition is shown in below Figures[13~16]. For the detailed analysis of the performance under various model mismatches and disturbances, refer to Section 6.

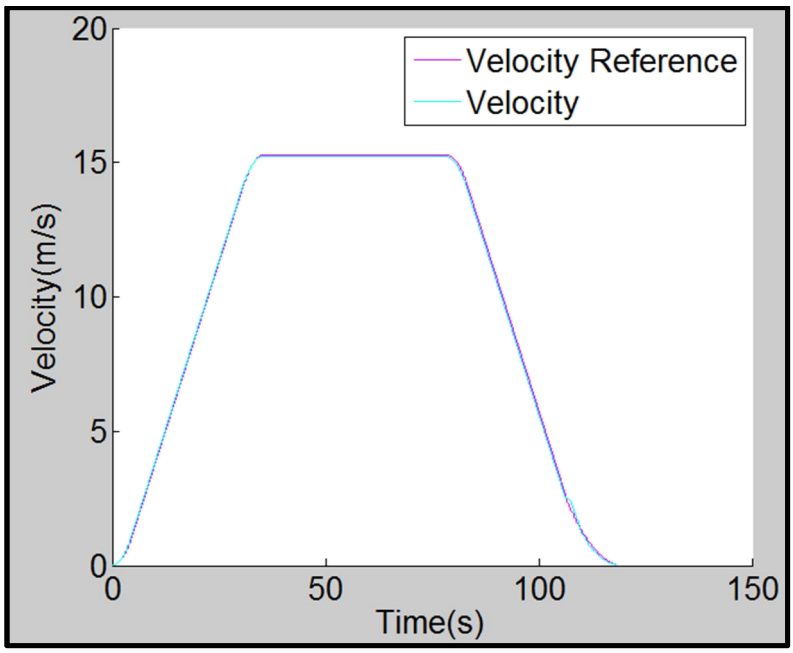


Figure 13 Velocity Tracking Performance

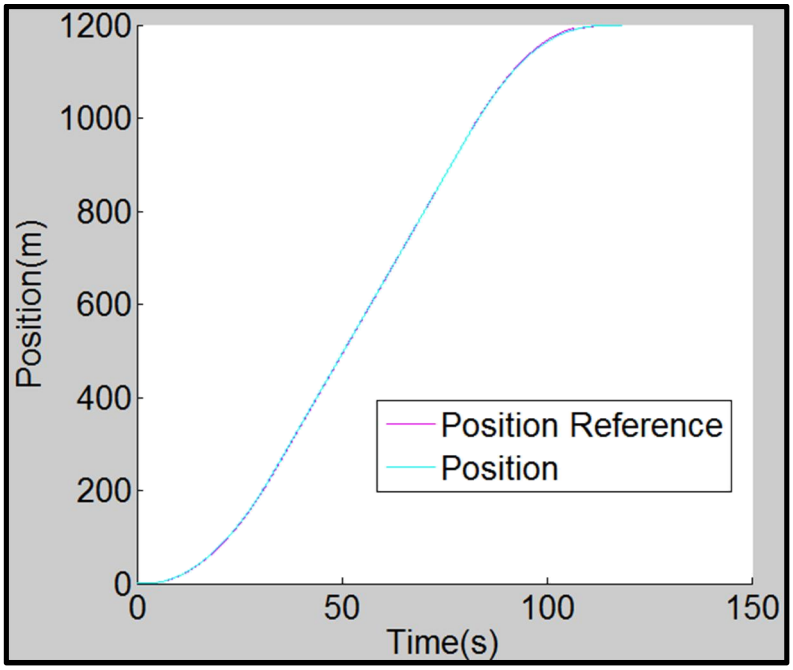


Figure 14 Position Tracking Performance

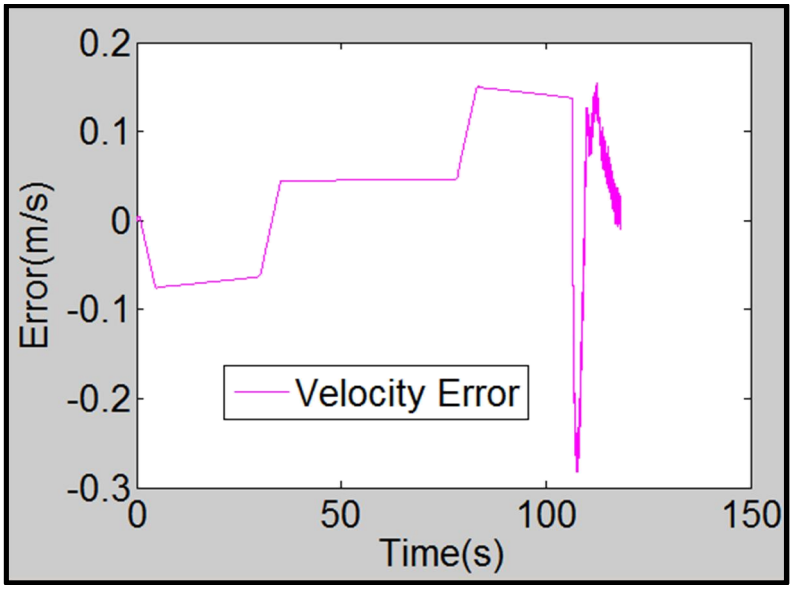


Figure 15 Velocity Tracking Error

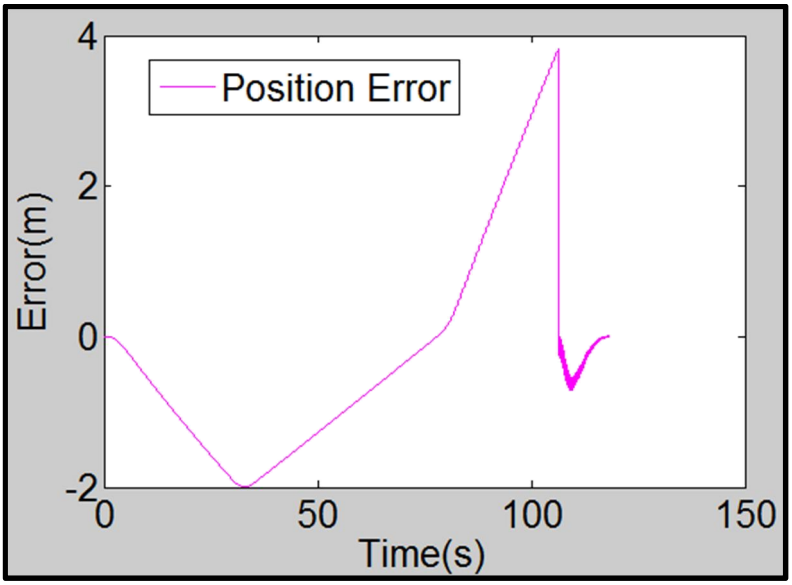


Figure 16 Position Tracking Error

4.2 Reference Generation

The first part where 2DOF controller is implemented only track velocity reference. The velocity reference for the first part is generated as follows:

Interval 1: Train starts from resting position and accelerates for five seconds. Second order function of time is used to generate velocity reference for smooth transition of reference to linear reference of next interval.

Interval 2: When previous reference has a slope of 1/2, a linear reference is generated for gradual increase of speed.

Interval 3: A second order function of time is introduce for smooth transition between ramp reference of Interval 2 and constant reference of Interval 4

Interval 4: Train cruises on set maximum velocity

Interval 5: Train starts slowing down. A second order function of time is introduce for smooth transition between constant reference of Interval 4 and ramp reference of Interval 6

Interval 6: When previous reference has a slope of -1/2, a linear reference is generated for gradual increase of speed.

Interval 7: A second order function of time is used to generate velocity reference for smooth stop of the train.

The velocity reference is made under the consideration of the jerk limit J_{max}. Here, jerk is defined as a second derivative of velocity respect to time. If velocity reference is

a first order polynomial function of time, the jerk reference is set at zero. If the velocity reference is a second order polynomial of time, the jerk reference would be a constant value. In specific, doubling the coefficient of the second order term would give the jerk reference. If the velocity reference $\mathrm{v_r(t)}$ is given as,

$$\mathrm{v_r(t)} = \mathrm{a_2}t^2 + a_1 t + a_0, \tag{4.1}$$

then the jerk reference $\mathrm{J_r}$ would be,

$$\mathrm{J_r} = 2\mathrm{a_2}, \tag{4.2}$$

for all t. The $\mathrm{a_2}$ value is chosen so that $2\mathrm{a_2}$ will be much smaller than $\mathrm{J_{max}}$. Throughout simulations, an $\mathrm{a_2}$ value of 0.05 is used while the unit of $\mathrm{v_r(t)}$ is set as meters. Given $\mathrm{J_{max}}$ set by the Korean Railroad standard is $0.8\mathrm{m/s^3}$, it is assumed to be a reasonably small value of $\mathrm{a_2}$.

The position reference is created by integrating the velocity reference. The velocity reference is shaped in a way that resulting position at the end of 2DOF controller interval would fall within PSM3 and PSM4, so that the velocity will not remain low for too long.

The second part where Receding Horizon Controller is implemented tracks both velocity and position reference. Velocity reference is assumed to be linear and position reference is driven from the integration of velocity reference with respect to time.

The velocity reference for LRHC is made as a second order polynomial function of time, and initial velocity $\mathrm{v_i}$ and the distance to final stop $\mathrm{d_s}$ is going to be given at the end of 2DOF controller interval. After algebraic manipulations, the amount of time to final stop t_f is then given as

$$\mathrm{t_f} = \frac{3\mathrm{d_s}}{v_i} \tag{4.3}$$

and velocity reference is given as

$$\mathrm{v_r(t)} = \mathrm{a_2}(t - t_f)^2, \tag{4.4}$$

where

$$\mathrm{a_2} = \frac{v_i}{(t_f)^2}. \tag{4.5}$$

The position reference of LRHC interval is derived from the velocity reference by integration respect to time.

The equations for the reference trajectories are given below:

For 2DOF controller

Interval 1

$$\mathrm{v_r(t)} = \frac{1}{20}t^2 \tag{4.6}$$

Interval 2

$$\mathrm{v_r(t)} = \frac{1}{2}t + v(t_1) \tag{4.7}$$

Interval 3

$$\mathrm{v_r(t)} = -\frac{1}{20}(t-5)^2 + v_{\max} \tag{4.8}$$

Interval 4

$$\mathrm{v_r(t)} = v_{\max} \tag{4.9}$$

Interval 5

$$\mathrm{v_r(t)} = -\frac{1}{20}t^2 + v_{\max} \tag{4.10}$$

Interval 6

$$\mathrm{v_r(t)} = -\frac{1}{2}t + v(t_5) \tag{4.11}$$

Interval 7

$$\mathrm{v_r(t)} = \frac{1}{20}(t-5)^2 \tag{4.12}$$

where t_i stands for the time at the end of i^{th} interval.

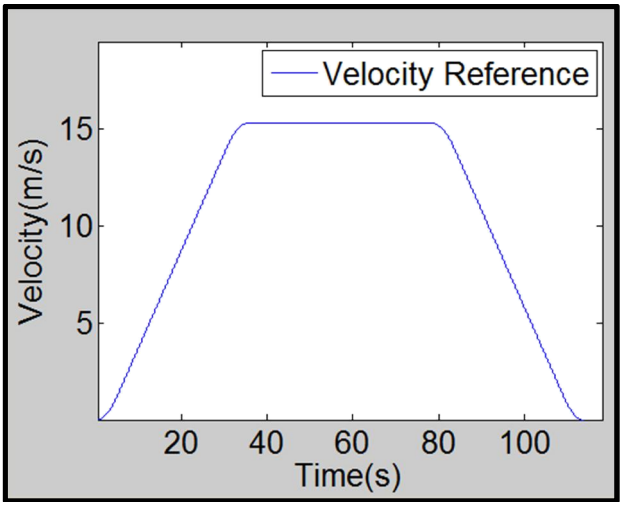


Figure 17 Velocity Reference for 2DOF Controller Interval

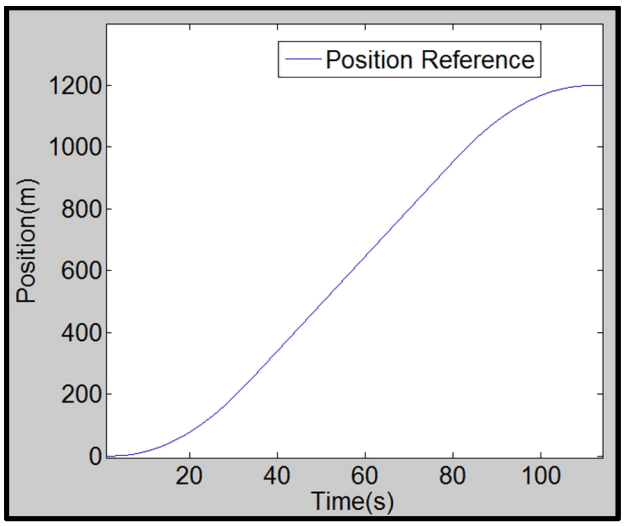


Figure 18 Position Reference for 2DOF Controller Interval

For Linear Receding Horizon Controller

Velocity

$$\mathrm{v_r(t)} = \mathrm{a_2}(t - t_f)^2 \tag{4.13}$$

Position

$$p_\mathrm{r}(\mathrm{t}) = \mathrm{a_2}(\tfrac{1}{3}t^3 - t_f t^2 + t_f^2 t) \tag{4.14}$$

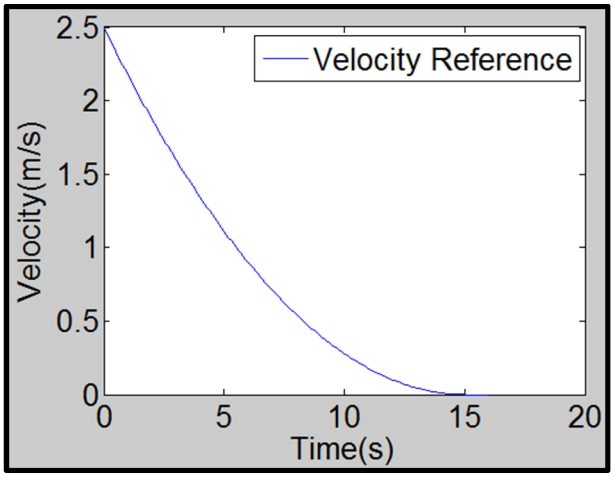


Figure 19 Velocity Reference of LRHC Interval

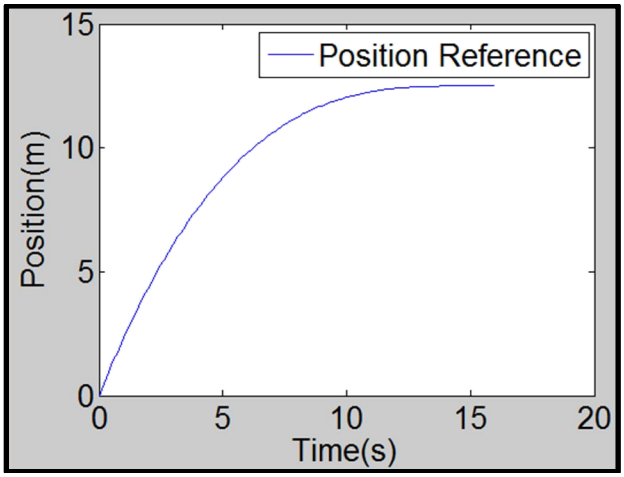


Figure 20 Position Reference of LRHC Interval

The train will generate a fresh reference for LRHC interval when the train slows to reach velocity of 2.5m/s. The position and velocity references are therefore adjusted. The final result of reference generation is shown in Figure 21 and 22.

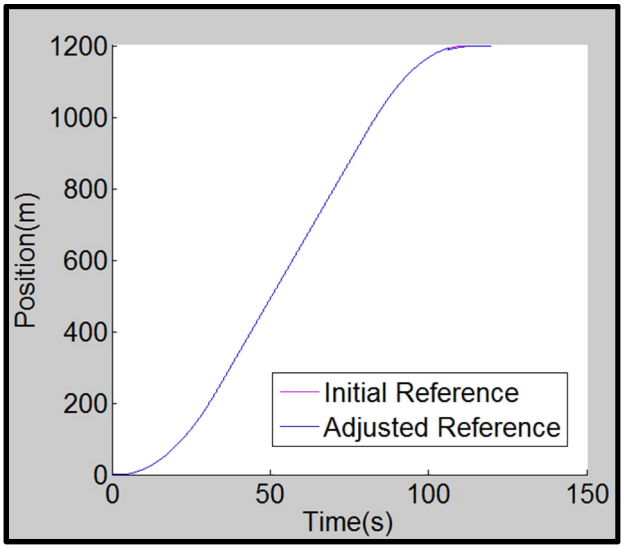


Figure 21 Adjusted Position Reference

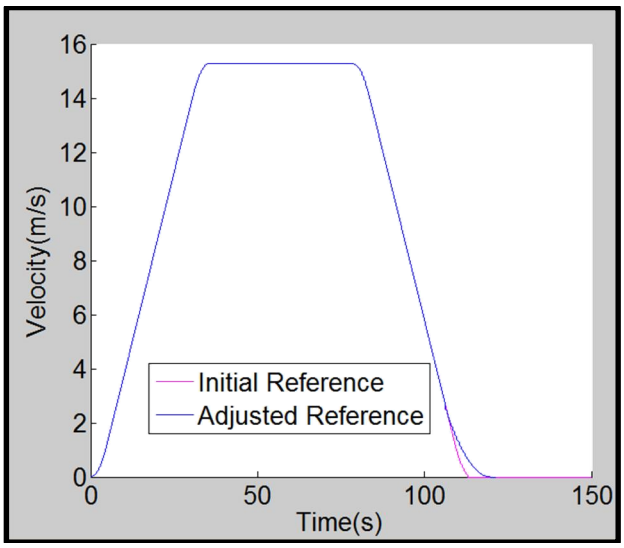


Figure 22 Adjusted Velocity Reference

4.3 Controllers

4.3.1 2DOF controller(Feedforward + Proportional-Integral Controller) [27]

4.3.1.1 Feedforward

Since the dynamics of actuators and plant along with velocity reference is all known, an appropriate feedforward input could be calculated. First, the velocity reference was shifted to match the delay caused by dead time and the transient response of actuators. Since the traction motor, regenerative brake, and air brake all exhibit nonlinearity respect to speed, it is not possible to calculate exact value for shifted time. From trial and error, a value of 1 second is found to be reasonable value for time shift. The simple feedforward equation is given as below:

$$\mathrm{u}_{\mathrm{i}}^{ff}(k) = \frac{m_i\left(v_r\left(k+\mathrm{ceil}\left(\frac{\text{time shift}}{Ts}\right)\right)-v_r(k)\right)}{Ts}, \tag{4.15}$$

where ceil(.) is a ceiling function and *Ts* is a sampling time.

4.3.1.2 PI

Since the feedforward input is in general not enough by itself unless model is perfect and disturbance is nonexistent, a PI controller is implemented in addition. From trial-and-error, proportional gain of 0.5 and integral gain of 2 is selected.

The schematics of 2DOF controller simulation is shown below in [Figure 23].

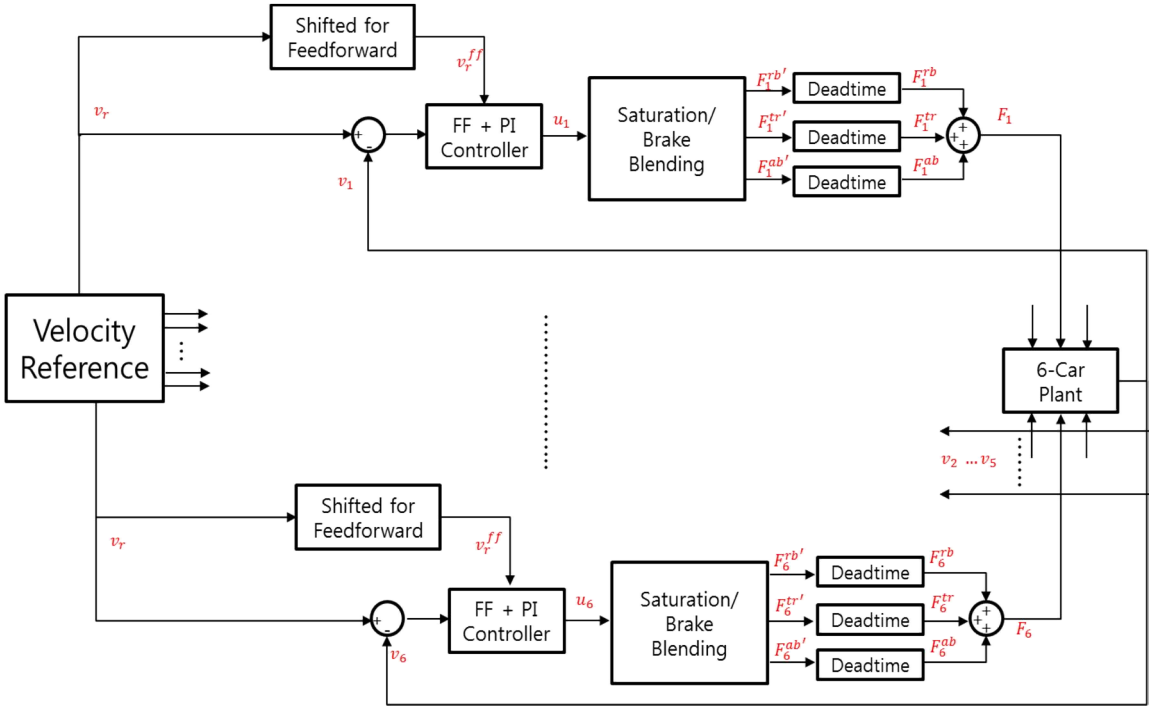


Figure 23 2DOF Controller Implementation Schematics

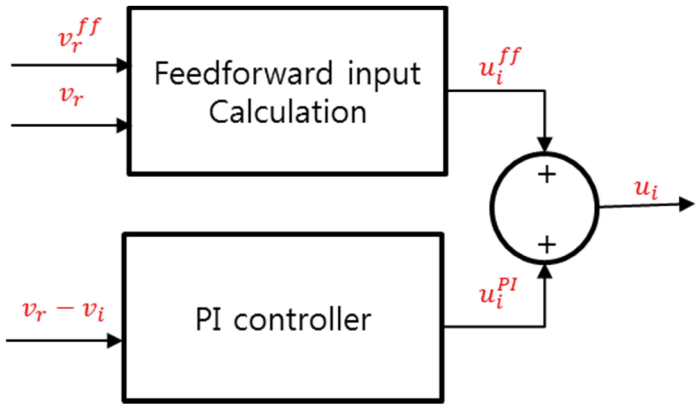


Figure 24 FF+PI Controller Block

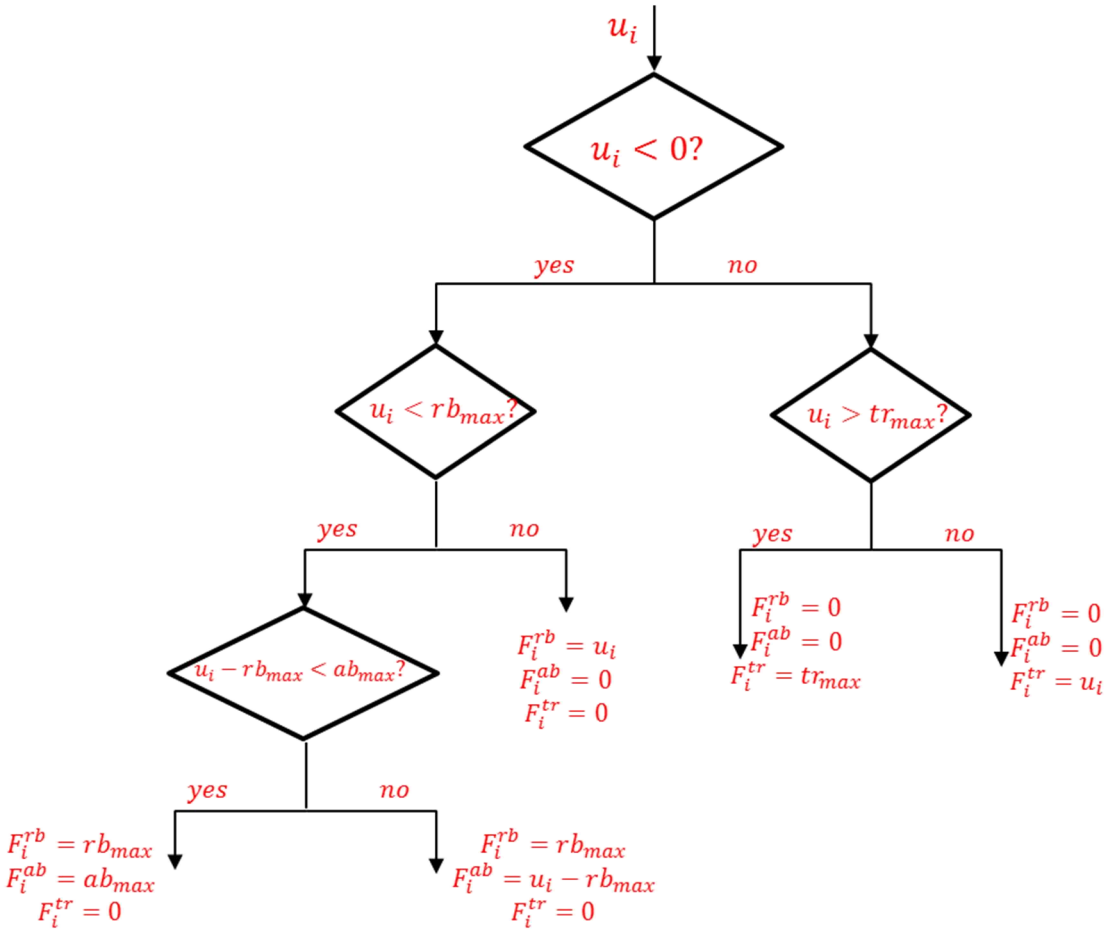


Figure 25 Saturation and Blending Block

4.3.2 Receding Horizon Controller

4.3.2.1. Introduction of Receding Horizon Control

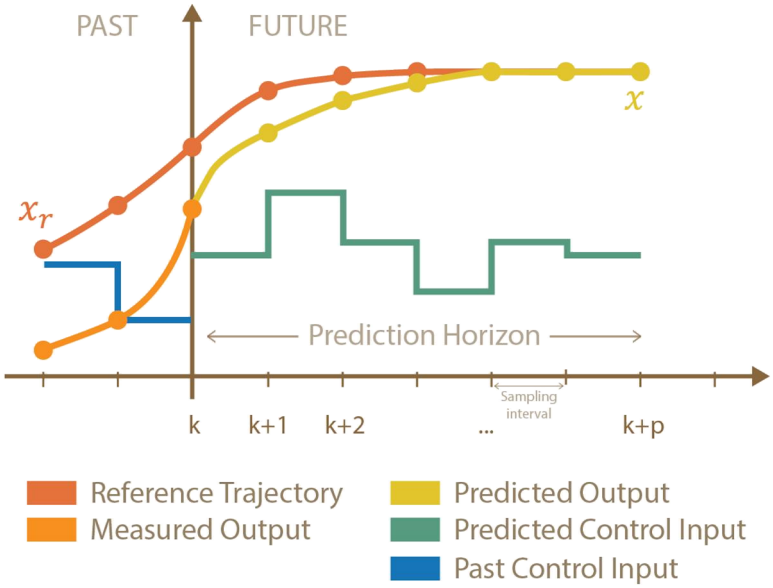


Figure 26 Receding Horizon Control

Receding horizon control, or more commonly known as Model Predictive Control, is an optimal control strategy with finite prediction horizon. Due to the computational overhead, it was originally used mainly on slow-moving plants[28]. The development of computer hardware and the field of optimization in past few decades enabled far wider use of RHC, even on fast-moving plant such as aerial vehicle[29]. It shows superb performance with accurate model and show more robustness compare to other infinite horizon optimal control, such as Linear Quadratic Control[28]. The down side is that the performance could still be significantly degraded when model in use is inaccurate[28]. Also for fast moving system, implementation could be difficult due to computational limits when dealing with nonlinear models or even linear models with high state dimensions. To implement RHC suitable for the metro train, two major difficulties had to be alleviated. First problem was creating a model which is accurate enough to capture the important dynamics of train but at the same time simple enough for control purpose. An engineering heuristics is employed to linearize the model with minimum discrepancies against the real plant; only the low velocity interval near the complete stop need to be considered as a result. Second, even when the model is simplified by linearization, state dimension had to be reduced in consideration of computational efficiencies in order to meet the real-time requirement of train controller. By taking advantage of the new train design, we could greatly reduce the state dimension. Instead of using one controller controlling the entire train, we designed a separate controller for each car. The justification is that if each individual cars track reference well, coupler dynamics would be small enough to be mitigated by controllers.

4.3.2.2 Control strategy

As previously mentioned, model mismatch degrades the performance of Receding Horizon Controller. As an effort to reduce the model uncertainty, the air brake is exclusively used for the control of train on the RHC interval. This measure was taken due to two reasons. First, the traction motor and the regenerative brake performance measurement raw data was not reliable under low speed. Second, air brake does not show nonlinearity related to velocity under low speed, therefore enable the easy implementation of Linear Receding Horizon Control. LRHC can be converted to standard Linear Constraint Quadratic Programming, which can be solved efficiently by convex optimization solvers. Moreover, by not using unreliable data and paying close attention to capture important dynamics of the train, the model can be assumed to be accurate.

4.3.3 Implementation

4.3.3.1. Air brake

In order to implement LRHC, linearization of air brake dynamics is necessary. The air brake dynamics, as discussed in Chapter III, displays transient response preceded by deadtime. In order to take account of deadtime efficiently, a first order Pade's approximation is applied on the transient response of the air brake[Figure 27]. The resulting transfer function is given as below:

$$H_{ab}(\mathrm{s}) = \frac{-5.29\mathrm{s}+52.9}{s^3+14.6s^2+51.29\mathrm{s}+52.9} \tag{4.16}$$

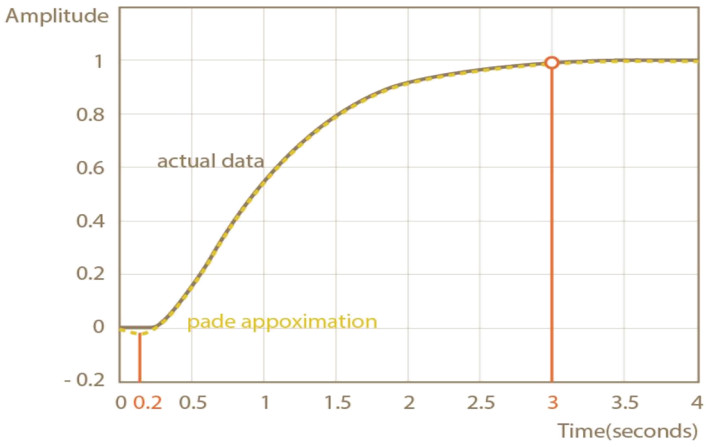
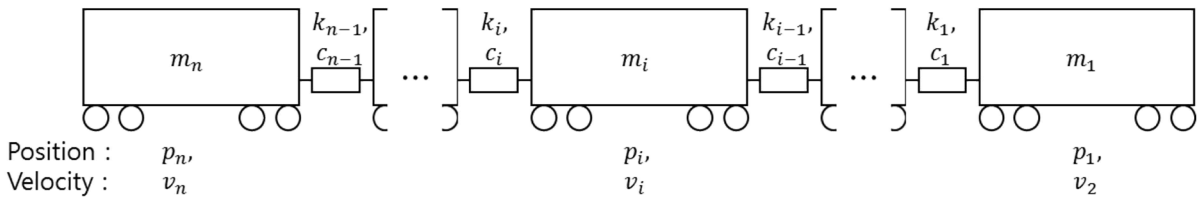


Figure 27 Air Brake Transient Response Approximation

4.3.3.2. Separate controllers

Even with a simplified train dynamics without the consideration of regenerative brake, traction motor and the deadtime, a 6-car train dynamics, written in a state-space representation form, is as below:



$$\dot{x} = Ax + B\begin{bmatrix} F_1 \\ \vdots \\ F_6 \end{bmatrix},\ y = Cx,\ x = \begin{bmatrix} p_1 \\ v_1 \\ \vdots \\ p_6 \\ v_6 \end{bmatrix}$$

$$A = \begin{bmatrix} R_{b1} & R_{c1} & 0_{2\times2} & 0_{2\times2} & 0_{2\times2} & 0_{2\times2} \\ R_{a2} & R_{b2} & R_{c2} & 0_{2\times2} & 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & R_{a3} & R_{b3} & R_{c3} & 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & R_{a4} & R_{b4} & R_{c4} & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times2} & R_{a5} & R_{b5} & R_{c5} \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times2} & 0_{2\times2} & R_{a6} & R_{b6} \end{bmatrix} \tag{4.17}$$

$$R_{ai} = \begin{bmatrix} 0 & 0 \\ \frac{k_{i-1}}{m_i} & \frac{c_{i-1}}{m_i} \end{bmatrix},\ R_{bi} = \begin{bmatrix} 0 & 0 \\ \frac{-k_{i-1}-k_i}{m_i} & \frac{-c_{i-1}-c_i}{m_i} \end{bmatrix}, R_{ci} = \begin{bmatrix} 0 & 0 \\ \frac{k_i}{m_i} & \frac{c_i}{m_i} \end{bmatrix} \tag{4.18}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{m_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{m_4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{m_5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{m_6} \end{bmatrix} \tag{4.19}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.20}$$

where k_i, and c_i are ith connector's spring constant and damping constant. $F_i, m_i, p_i,$ and v_i are the force, mass, position, and velocity of the ith car.

Even with a moderate horizon window size, the dimension of resulting LCQP problem dimension is quite large, making it difficult to implement in real time under hardware with reasonable cost. To mitigate this problem, a separate, six 1-car controllers are implemented. [Figure 28]

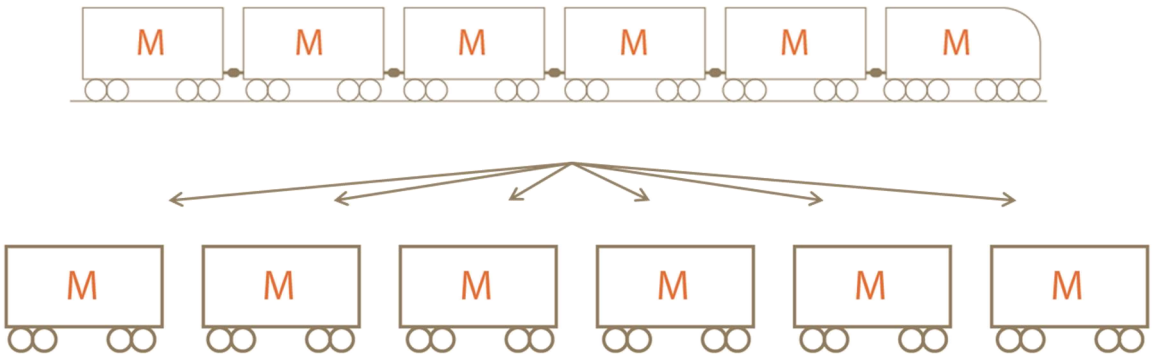


Figure 28 Separate Controllers

The justification of this approach is provided below:

1. Every vehicle of the new train model in consideration is M-car, so it is technically possible to implement separate controllers on each vehicle.
2. The existence of coupler dynamics is the only difference between 1-car model and 6-car model. If each car tracks both the velocity and position well, the coupler dynamics would be negligible.

The resulting model of 1-car train, in continuous time is given below:

$$\dot{\hat{x}}_i = \hat{A}_i\hat{x}_i + \hat{B}_i\mathrm{u_i},\ \ \hat{y}_i = \hat{C}_i\hat{x}_i,\ \ \hat{x}_i = \begin{bmatrix} p_i \\ v_i \\ \zeta_{i1} \\ \zeta_{i2} \\ \zeta_{i3} \end{bmatrix}$$

$$\hat{A}_i = \begin{bmatrix} A_i^o & \mathrm{B_i^o} * C_i^{ab} \\ \mathrm{O}_{3\times 2} & \mathrm{A_i^{ab}} \end{bmatrix} \tag{4.21}$$

$$A_i^o = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},\ B_i^o = \begin{bmatrix} 0 \\ \frac{1}{m_i} \end{bmatrix} \tag{4.22}$$

$$\hat{B}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.23}$$

$$\hat{C}_i = [0\ 1\ 0\ 0\ 0], \tag{4.24}$$

where p_i and v_i are position and velocity of ith vehicle. ζ_{i1}, ζ_{i2}, and ζ_{i3} are dummy states resulting from the Pade's approximation of air brake dynamics.

The model used in LRHC is given below:

$$\hat{x}_i(\mathrm{k}) = \widehat{A_i^d}\hat{x}_i(\mathrm{k}-1) + \widehat{B_i^d}\mathrm{u_i}(\mathrm{k}-1),\ \ \hat{y}_i(\mathrm{k}) = \widehat{C_i^d}\hat{x}_i(k),\ \ \hat{x}_i(k) = \begin{bmatrix} p_i(k) \\ v_i(k) \\ \zeta_{i1}(k) \\ \zeta_{i2}(k) \\ \zeta_{i3}(k) \end{bmatrix} \tag{4.25}$$

where $\widehat{A_i^d}$, $\widehat{B_i^d}$, and $\widehat{C_i^d}$ are corresponding matrices from the discretization of 1-car continuous time model by MATLAB c2d command.

4.3.3.3. Conversion to Linear Constraint Quadratic Programming

A standard Linear Constraint Quadratic Programming could be written as below:

Minimize

$$\frac{1}{2}\theta^{\mathrm{T}}\mathrm{H}\theta + \mathrm{f}^{\mathrm{T}}\theta \tag{4.26}$$

subject to

$$\mathrm{C}\theta \leq \mathrm{b}$$
$$\mathrm{E}\theta = \mathrm{d}$$

A linear, discrete time Receding Horizon servomechanism problem with input and state constraints can be formulated as below:

$$\mathrm{argmin}_{\mathrm{u(j|k)}} \sum_{\mathrm{j=k}}^{\mathrm{k+p+1}} \left(\mathrm{x_{r(j)}} - x_{(j|k)}\right)^{T} Q_{(j|k)} \left(\mathrm{x_{r(j)}} - x_{(j|k)}\right) + u_{(j|k)}^{T} R_{(j|k)} u_{(j|k)} \tag{4.27}$$

subject to

$$\mathrm{x(k+1)} = \mathrm{Ax(k)} + \mathrm{Bu(k)} \tag{4.28}$$
$$-\beta \leq \mathrm{x_k} \leq \alpha$$
$$-\delta \leq \mathrm{u_k} \leq \gamma$$

where $\mathrm{A} \in \mathrm{R^{n\times n}}, \mathrm{B} \in \mathrm{R^{n\times m}}, \mathrm{x} \in \mathrm{R^{n\times 1}}$, and $\mathrm{u} \in \mathrm{R^{m\times 1}}$. p is the size of horizon window, $\mathrm{x_{r(j)}}$ is the reference at time t=j, and $x_{(j|k)}$ and $u_{(j|k)}$ are predicted $x_{(j)}$ and $u_{(j)}$.

This could be converted to LCQP problem by setting

$$\theta = [\mathrm{x}_{(\mathrm{k|k})}^{T} \cdots x_{(k+p+1|k)}^{T}\ \mathrm{u}_{(\mathrm{k|k})}^{T} \cdots u_{(k+p|k)}^{T}] \tag{4.29}$$

$$\mathrm{H} = \mathrm{diag}[\mathrm{Q_{(k|k)}}| \cdots |\mathrm{Q_{(k+p+1|k)}}|\mathrm{R_{(k|k)}}| \cdots |\mathrm{R_{(k+p|k)}}] \tag{4.30}$$

$$\mathrm{f}^{\mathrm{T}} = -\left[\mathrm{x}_{\mathrm{r(k)}}^{T} \cdots x_{r(k+p+1)}^{T}\ 0 \cdots 0\right]^{\mathrm{T}}\mathrm{H} \tag{4.31}$$

$$C=\begin{bmatrix} I_{n(p+1)} & 0_{mp} \\ -I_{n(p+1)} & 0_{mp} \\ 0_{n(p+1)} & I_{mp} \\ 0_{n(p+1)} & -I_{mp} \end{bmatrix} \tag{4.32}$$

$$b = [\alpha^T \beta^T \gamma^T \delta^T]^T \tag{4.33}$$

$$E=\begin{bmatrix} I_n & 0_n & \cdots & \cdots & \cdots & \cdots & 0_n \\ A & -I_n & 0_n & \cdots & B & 0_n & \cdots \\ 0_n & A & -I_n & 0_n & \cdots & B & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4.34}$$

$$d = \left[x_{(k|k)} \ 0 \cdots 0\right]^T, \tag{4.35}$$

where I_l and O_l are $l \times l$ identity matrix and zero matrix, and C, E, θ, f^T, b and d are matrices and vectors of appropriate dimensions. Note that when j=k, $x_{(j|k)}$ is a real measured state at time k, which makes RHC a kind of feedback control.

4.3.3.4. Actual Implementation

The actual implementation of LRHC is given in below pseudo code

SET LRHC sampling time, number of horizon windows used, train mass, passenger mass, assumed value of deadtime, ω_n value of air brake transient response in standard second order transfer function, coupler's spring constant, coupler's damping constant to appropriate parameters
READ distance between previous station to stopping station, current distance from previous station, current velocity, real train model
GENERATE velocity and position reference, 1-car train models used in LRHC
INITIALIZE θ, f, H, C, b, E, d
FOR each sampling instance
READ position reference, velocity reference, position, velocity, previous result of quadratic programming
GENERATE θ, f, H, C, b, E, d
COMPUTE quadprog(θ, f, H, C, b, E, d)
SET input force from the result of quadprog
OUTPUT input force
ENDFOR

For further detail, refer to the code in Appendix.

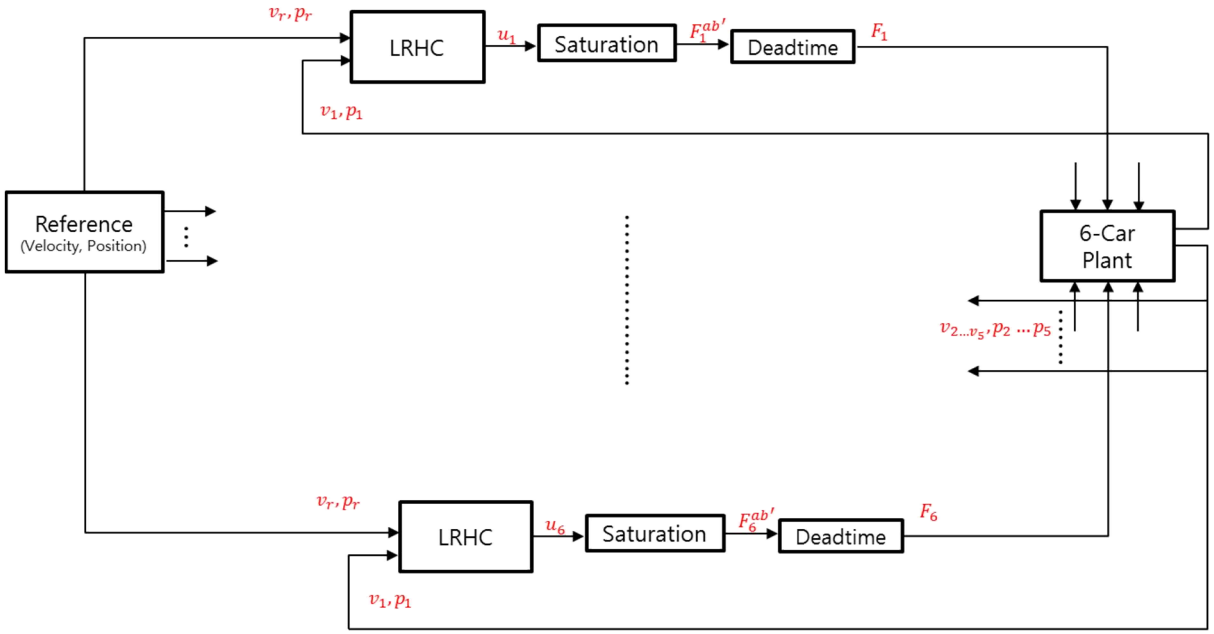


Figure 29 Proposed LRHC Implementation Schematics

5 Simulation

The simulations conducted to test the performance of controller are presented here, along with the explanation of the metro train precision stop simulator code included in Appendix.

5.1 Simulation

5.1.1. Test for robust stability and performance

Since the stability of the LRHC under deadtime phenomenon along with variations in model parameters coupled with disturbances is not trivial, a brute-force method is employed to analyze the stability of the controller. The mass of train, ω_n values from the standard second order transfer function of actuator transient responses, and the amount of deadtime of actuators are varied from -15% to +15% of the original value with a fixed increment of 1%. All possible combinations given above condition are tested. The proposed two-stage controller performed well under semi-nominal condition and since this thesis is on precision stop, the simulation is carried out exclusively on LRHC interval. The assumption here is that if all of the final stop position error falls within the specification ($\pm$0.1m) and at the same time all of the test results pass an outlier detection test, it would be reasonable to believe that there are strong possibility that the proposed controller satisfies robust stability and performance, which could serve as a basis for further detailed simulation and/or analytical work.

5.1.2 Test for performance under different horizon window size and controller output frequency

Performance of the final stop position error by LRHC controller is tested under varying horizon window size and controller output frequency. Assuming realistic computational limitation, reasonable values of horizon window size and controller output frequency are explored.

5.1.3 Test for the effect of reference shape on stop error

The effect of employing different means of reference generation is tested. Linear and quadratic functions of time as velocity references are compared.

5.2 Implementation details

The simulation is carried out under MATLAB. Everything is implemented in code (.m file) except the data table of the interpolated nonlinear function for actuator dynamics. The code is structured as follows.

- Variation Interval
- Initialization
- Reference Generation
- Plant Model Generation
- 2DOF Controller Implementation
- Linear Receding Horizon Controller Implementation
- Plot
- Output

5.2.1. Variation Interval/Disturbance Generation

The mass of train, ω_n values from the standard second order transfer function of actuator transient responses, and the amount of deadtime of each actuators can be varied multiple time with set intervals. Zero-mean Gaussian noise is generated to be used for measurement noise and actuator disturbance.

5.2.2. Initialization

Initialize various parameters used during simulation. Also sampling time, train mass, and actuator dynamics can be adjusted. Constrains on actuators and plant is determined here.

5.2.3. Reference Generation

Velocity and position reference is generated. Two separate reference is generated for the interval from start to velocity above 2.5m/s and below 2.5m/s to the stop.

5.2.4. Plant Model Generation

Based on the supplied parameters, a discretized plant model of 6-car train is generated. Default sampling time is set at 1000Hz.

5.2.5. 2DOF Controller Implementation

A combination of feedforward and PI controller is implemented to track the reference until the train arrives at the LRHC interval. Brake blending is considered.

5.2.6. Linear Receding Horizon Controller Implementation

Six separate 1-car controllers are implemented. Each controller has plant model incorporating the Pade approximated air brake dynamics the deadtime.

5.2.7. Plot

A short summary of the test result is plotted. Position and position reference, and velocity and velocity reference of all six cars are plotted. Position error and velocity error of all six cars respect to time are shown.

5.2.8. Output

The result of each iteration representing controller performance (final stop error) is recorded to a .csv file. The variable values for each loop are also recorded to .mat file.

6 Result

This chapter provides the detailed analysis of the simulation outlined in

6.1 Test for robust stability and robust performance

The mass of train, ω_n values from the standard second order transfer function of actuator transient responses, and the amount of deadtime of actuators are varied from -15% to +15% of the original value with a fixed increment of 1%[Figure 30]. All possible combinations given above condition are recorded in data along with the final stop position error. Also, the combination of extreme and middle values of ±20% variation on all three parameters also resulted in satisfactory performance [Figure 31~33].

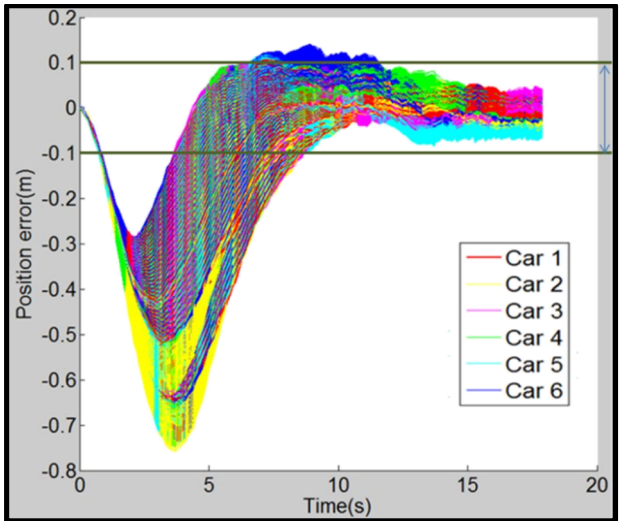


Figure 30 RHC tested -15~15percent, 1 percent interval with measurement noise and actuator disturbance

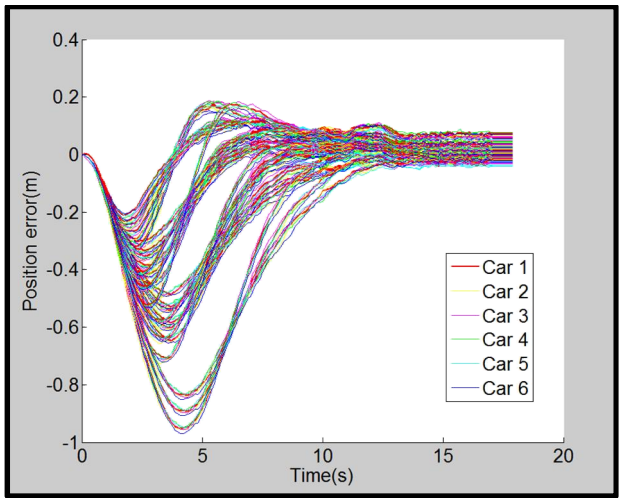


Figure 31 RHC tested -20~20percent, 20 percent interval with measurement noise and actuator disturbance

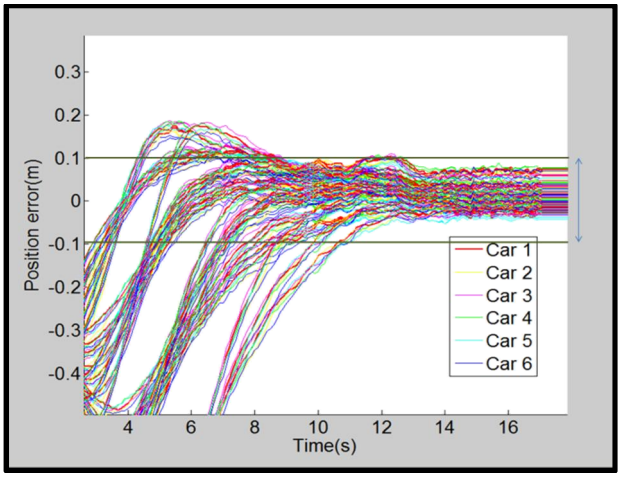


Figure 32 RHC tested -20~20percent, 20 percent interval with measurement noise and actuator disturbance

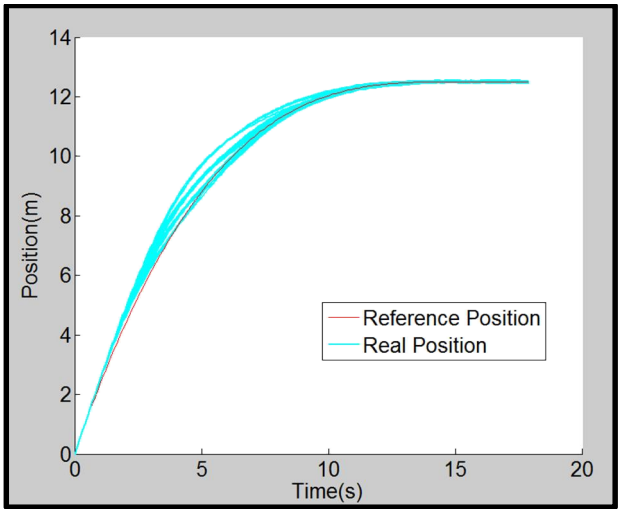


Figure 33 RHC tested -20~20percent, 20 percent interval with measurement noise and actuator disturbance

6.2 Test for performance under different horizon window size and different controller output frequency

Performance of the final stop position error by LRHC controller is tested under horizon window size of 1 to 30 and the frequency of window 1 to 30, without the presence of any disturbances. It is reasonable to believe that the total prediction window should cover fair portion of air brake's transient response. If not so, the controller will not be able to properly capture the effect of current input command that happens after the prediction window. Given the rise time of the air brake from experimental data is roughly 2seconds, an adequate total prediction window size is should be around 2seconds. The result [Figure 34] validates this hypothesis.

In Figure 34, the final stop error performances are drastically degraded when horizon frequency is below 10Hz and number of windows are below 20. Therefore, controller sampling time of 10Hz and horizon window size of 20 would be reasonable choice when implementing LRHC for the train model. The result shows finer control output frequency performs better than coarse control output frequency, and also the more number of horizon window performs better then lesser number of horizon window in general.

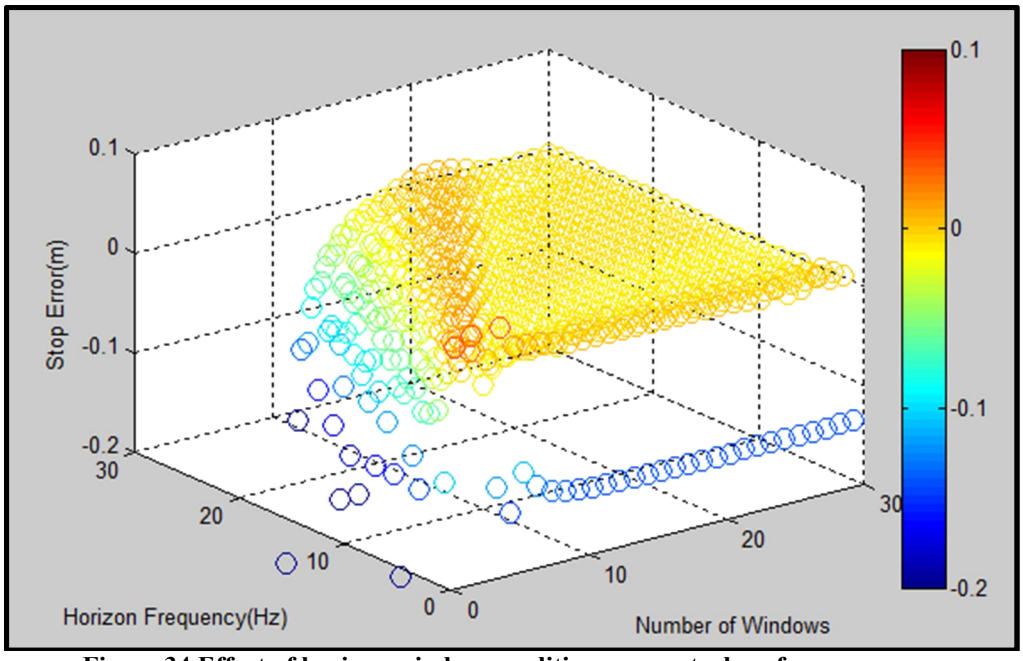


Figure 34 Effect of horizon window conditions on control performance

6.3 Test for the effect of reference shape on stop error

The velocity reference for proposed controller is generated when the train speed reaches 2.5m/s. The control performance between velocity reference of a first order and a second order polynomial of time is compared.

The initial velocity is 2.5m/s, and 12.5 meters are left before the stopping position. Zero mean Gaussian noise with standard deviation of 0.33 percent is introduced as variation on measurement and actuation. 15 percent variation on train mass, ω_n value of the transient response of air brake in standard second order transfer function, and the amount of deadtime is introduced. The train is to stop before 17 seconds after the LRHC interval starts.

When the velocity reference is a first order polynomial of time, the train fails to stop within ± 0.1m of predetermined position[Figure 35]. However, when the velocity reference of a second order polynomial of time is utilized, the train stopped well within the ± 0.1m error boundary[Figure 36]. Therefore, it is favorable to use second order

polynomial of time as a velocity reference for LRHC interval.

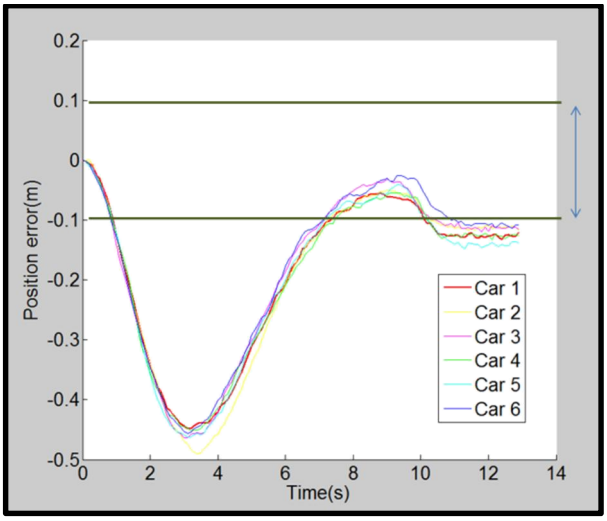


Figure 35 15 percent variations, linear reference

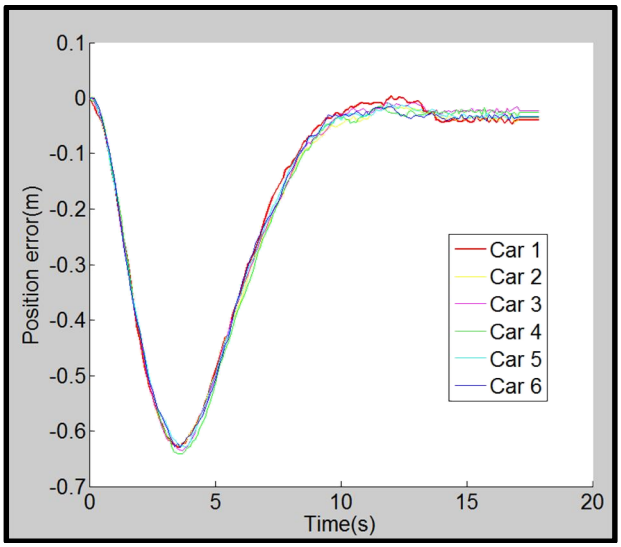


Figure 36 15 percent variations, quadratic reference

7 Conclusion

7.1 Summary

The thesis began with an introduction to metropolitan train system, so unfamiliar readers would have understanding of metro train stop process. Modeling and controller implementation was then discussed. For modeling, consideration of deadtime was one of the most important factors. For controller implementation, exclusively using air brake enabled use of LRHC, which shows good performance for the precision stop purpose. Other factors that may influence the performance of the proposed LRHC were examined. First was the parameters related to horizon window. In general, the higher the horizon frequency and the number of window utilized, the better were the precision stop performances. However, when total prediction window fell below an adequate amount (2 second) to cover transient response of actuator, the controller performed very poorly. Controller frequency of 10Hz and window size of 20 was found to be reasonable. Second was the shape of the velocity reference. When the velocity reference was a second order polynomial of time, the proposed controller performed better than the case when the velocity reference was a first order polynomial of time. Third, the robustness of the LRHC under model mismatch, measurement noise, and disturbance on actuators are also tested. The proposed LRHC successfully brought train to stop within ± 0.1m error boundary under 15 percent deviations of ω_n value of the air brake's transient response in standard form of second order transfer function, the amount of deadtime, and the train mass. Measurement noise and actuator disturbance were also introduced. Afterward, the simulation to test robust stability and performance were explained in detail. Analysis was then applied on the data collected from the simulation. Robust stability and performance were tested empirically.

7.2 Future Work

The model is validated because the modeling parameters are based upon real data supplied by KRRI. However, there was limitation on controller efficiency validation because the test procedure was too costly and time consuming. When appropriate test bed is available, test on real model would provide deeper insight on train controller implementation

When the efficiency of LRHC is thoroughly tested, the application of offset-free Receding Horizon Control could be interesting in case of malicious attack and/or fault conditions. Such research would eventually be necessary because the metropolitan train is widely used public transportation, which requires very high safety standard.

7.3 Concluding Remark

The new train will be consisted of 6 M-cars, along with the usual reliable air brake. This allows the proposed implementation of Linear Receding Horizon Control for precision stop.

Assuming that the train model used and sensors deployed in the train are reasonably accurate because metropolitan train is a public transportation system with high safety standard, the scope of the test on controller robustness under simulation goes far beyond the reasonable amount. Therefore, it is reasonable to believe that the proposed controller may be reliable enough to be deployed, so the costly test on real train is justified.

Overall, the LRHC seems promising for precision stop purpose. With increased accuracy of modeling and computational power, even far more precise stop position control will be possible, which may become necessary in future.

Appendix

Metro Train Precision Stop Simulation Code

```
% Simulator for Precision Stop of Metropolitan Train with Receding Horizon
% Controller. By Sungmin Aum, 2015
%Easy use for testing variation on train mass, time delay of air brake,
%transient response of air brake, and coupler dynamics
clear;
fclose all;
delay_deviation_low=-10;
delay_deviation_high=10;
wn_deviation_low=-10;
wn_deviation_high=10;
mass_deviation_low=-10;
mass_deviation_high=10;
increment=0.01;

m_noise=0.01;
a_noise=0.01;
%noise and disturbance generation
measurement_noise=zeros(12,20000);
actuator_disturbance=zeros(6,20000);

for k_3=1:17000 % generate for 170 seconds. could be shortened
for i_9=1:12
measurement_noise(i_9,k_3)=normrnd(0,m_noise);
end
for i_10=1:6
actuator_disturbance(i_10,k_3)=normrnd(0,a_noise);
end
end

%file name
savefile=
sprintf('RHC_polytope_delay_%d_to_%d_wn_%d_to_%d_mass_%d_to_%d_inc_%
d_mn_%d_an_%d.csv',delay_deviation_low,delay_deviation_high,wn_deviation_low
,wn_deviation_high,mass_deviation_low,mass_deviation_high,increment,m_noise,a_
noise);

% set file stream
file_MM = fopen(savefile,'w');

for dd=delay_deviation_low:delay_deviation_high
      delay_dev=increment*dd;
  % 0.1 means real time delay is 10percent more than that of original
for wd=wn_deviation_low:wn_deviation_high
      wn_dev=increment*wd;
```

```
% transient response wn deviation 0.1 means 10 percent 'faster' response
for dm=mass_deviation_low:mass_deviation_high
      dev_m=increment*dm;


pTs=0.001;

fault_flag=0; % no fault
delay_dev=0.1;
wn_dev=0.1;
dev_m=0.1;
t_delay=0.2*(1+delay_dev); %model assumed pure delay of 0.2 second
delay_constant=ceil(t_delay/pTs);
position=[0 0 0 0 0 0]';
position_p=[0 0 0 0 0 0]'; %previous
count=0;
ab_count=[0 0 0 0 0 0]';rb_count=[0 0 0 0 0 0]';tr_count=[0 0 0 0 0 0]';
velocity=[0 0 0 0 0 0]';
velocity_p=[0 0 0 0 0 0]'; %previous
u_cont=[0 0 0 0 0 0]'; % controller input
u_cont_p=[0 0 0 0 0 0]'; % controller input
Force_d=[0 0 0 0 0 0]';
u_ff=[0 0 0 0 0 0]';
u_ff_brake_temp=u_ff;
u_ff_rb_tmp=u_ff;
u_ff_rb=u_ff;
u_ff_ab=u_ff;
deadtime=ceil(t_delay/pTs);
%
t_trsp=0:pTs:5;
rise_time=1;
a_value=6.9;
t_delay=0.2;
b_value=exp(t_delay*a_value);
trsp=-b_value*exp(1).^(-a_value*t_trsp)+1;
trsp(1:ceil(0.2/pTs))=0;

avg_force_ratio=-b_value*(-1/a_value)*(exp(-a_value*1)-exp(-
a_value*t_delay))/rise_time+(rise_time-0.2); % t from 0.2 to 1

max_vel=55/3.6;
reference_slope=1/2;


Force_m=[0 0 0 0 0 0]'; %traction from motors
Force_r=[0 0 0 0 0 0]'; %running resistance
Force_rb_max=[0 0 0 0 0 0]'; %vector
Force_ab_max=[0 0 0 0 0 0]'; %vector
```

```
Force_m_max=[0 0 0 0 0 0]'; %traction from motors
x1e=[0 0 0 0 0 0]';
x2e=[0 0 0 0 0 0]';
station_d=1200; %1200meters apart
mass_c=[42153 40463 40463 40463 40463 41990]'; %empty car
mass_p=[6000 3000 5500 2000 4000 3500]'; % passengers
mass_t=(mass_c+mass_p)*(1+dev_m);
psm1=546;
psm2=108.5;
psm3=21;
psm4=3.5;


delta_t=pTs;
kk=[3.6 3.6 3.6 3.6 3.6]*10^6';
cc=[8333    8333    8333    8333    8333]';


%reference generation
end_time=150;
t=0:pTs:end_time;
end_count=ceil(end_time/pTs);
x2r=t;
x1r=t;
five_sec=ceil(5/pTs);

 testr=0:pTs:end_time;
 five_second_int=0:pTs:5;
 testr=(1/10*reference_slope)*testr.^2; % t=5 sec has slope of 1/2

 int_1=ceil(5/pTs); % first interval before slope is 1/2
 int_v=testr(int_1);
 linear_int_dv=max_vel-2*int_v;    %velocity increment covered by linear interval
 linear_int=0:pTs:(1/reference_slope)*linear_int_dv;
 linear_int=(reference_slope)*linear_int+int_v;
 int_3=-(1/10*reference_slope)*(five_second_int-5).^2+(max_vel);
 int_4=-(1/10*reference_slope)*(five_second_int).^2+(max_vel);


 testr(int_1+1:int_1+length(linear_int))=linear_int;
 testr(int_1+length(linear_int)+1:2*int_1+length(linear_int)+1)=int_3;

ind=find(testr>max_vel);
end_first_interval=ind(1)-1;

for i=2:end_first_interval
      x2r(i)=testr(i);
      x1r(i)=x1r(i-1)+(x2r(i-1)+x2r(i))*pTs/2;
```

```
end
curve_intverval_distance=x1r(end_first_interval);
cruise_distance=station_d-2*curve_intverval_distance;
end_second_interval=ceil((cruise_distance/max_vel)/pTs)+end_first_interval;
for i=end_first_interval:end_second_interval
      x2r(i)=max_vel;
       x1r(i)=x1r(i-1)+(x2r(i-1)+x2r(i))*pTs/2;
end

 end_third_interval=end_second_interval+length(five_second_int);
 x2r(end_second_interval+1:end_third_interval)=int_4;
for i=end_second_interval+1:end_third_interval
       x1r(i)=x1r(i-1)+(x2r(i-1)+x2r(i))*pTs/2;
end
end_fourth_interval=end_third_interval+length(linear_int);

linear_int_2=0:pTs:(1/reference_slope)*linear_int_dv;
linear_int_2=-(reference_slope)*linear_int_2+x2r(end_third_interval);

x2r(end_third_interval+1:end_fourth_interval)=linear_int_2;
for i=end_third_interval+1:end_fourth_interval
       x1r(i)=x1r(i-1)+(x2r(i-1)+x2r(i))*pTs/2;
end
int_5=(1/10*reference_slope)*(five_second_int-5).^2;
end_fifth_interval=end_fourth_interval+length(int_5);
x2r(end_fourth_interval+1:end_fifth_interval)=int_5;

for i=end_fourth_interval+1:end_fifth_interval
       x1r(i)=x1r(i-1)+(x2r(i-1)+x2r(i))*pTs/2;
end

station_stop=find(x1r>1200,1);
% x2r=x2r-x2r(station_stop);
x1r(station_stop:end)=1200;
x2r(station_stop:end)=0;

srit=find(x2r(end_second_interval:end)<2.5,1);
start_rhc_interval=end_second_interval+srit;

%feedforward ref
x2r_ff=x2r(ceil(1/pTs):end);
x2r_sz=size(x2r);
x2r_ff_sz=size(x2r_ff);
x2r_ff=[x2r_ff zeros(1,x2r_sz(2)-x2r_ff_sz(2))];


x2r_ff_dead=x2r(ceil(t_delay/pTs):end);
x2r_ff_dead_sz=size(x2r_ff);
```

```
x2r_ff_dead=[x2r_ff zeros(1,x2r_sz(2)-x2r_ff_dead_sz(2))];


Force_rb=zeros(6,end_fifth_interval); %regenerative brake
Force_ab=zeros(6,end_fifth_interval); %air brake
Force_tr=zeros(6,end_fifth_interval); %traction
Force_t=zeros(6,end_fifth_interval); % total force
Force_d=zeros(6,end_fifth_interval); % total force


Force_rb_des=zeros(6,end_fifth_interval); % desired
Force_ab_des=zeros(6,end_fifth_interval); % desired
Force_tr_des=zeros(6,end_fifth_interval); % desired

x_ab=zeros(12,end_fifth_interval);
x_rb=zeros(12,end_fifth_interval);
x_tr=zeros(12,end_fifth_interval);


%control
gTable=load('tables_refenceVelocity,traction,brake_140610.mat');
%traction N,m/s
gTable_unt.trac_v=gTable.trac_v/3.6;
gTable_unt.trac_f=gTable.trac_f*1000;
%air brake N,m/s
gTable_unt.air_brake_v=gTable.air_brake_v/3.6;
gTable_unt.air_brake_f=gTable.air_brake_f*1000;
%disk brake N,m/s
gTable_unt.air_brake2_v=gTable.air_brake2_v/3.6;
gTable_unt.air_brake2_f=gTable.air_brake2_f*1000;
%regenerative brake N,m/s
gTable_unt.reg_brake_v=gTable.reg_brake_v/3.6;
gTable_unt.reg_brake_f=gTable.reg_brake_f*1000;


%transient

%airbrake
wn_ab=2.3;
n1_ab=[0 wn_ab^2];
d1_ab=[1 2*wn_ab wn_ab^2];
[Aab_,Bab_,Cab_,Dab_]=tf2ss(n1_ab,d1_ab);
mp1_ab=Aab_;mp2_ab=Bab_;mp3_ab=Cab_;mp4_ab=Dab_;
Aab_(1,:)=[0 1];Aab_(2,:)=[mp1_ab(1,2) mp1_ab(1,1)];
Bab_=[0;1];
Cab_=[mp3_ab(1,2) mp3_ab(1,1)];

s1_ab=ss(Aab_,Bab_,Cab_,Dab_);s2_ab=c2d(s1_ab,pTs);
```

```
Aab_d=s2_ab.a;
Bab_d=s2_ab.b;
Cab_d=s2_ab.c;

%air brake when it comes down
wn_ab_dn=wn_ab*5;
n1_ab_dn=[0 wn_ab_dn^2];
d1_ab_dn=[1 2*wn_ab_dn wn_ab_dn^2];
[Aab_dn,Bab_dn,Cab_dn,Dab_dn]=tf2ss(n1_ab_dn,d1_ab_dn);
mp1_ab_dn=Aab_dn;mp2_ab_dn=Bab_dn;mp3_ab_dn=Cab_dn;mp4_ab_dn=Dab_d
n;
Aab_dn(1,:)=[0 1];Aab_dn(2,:)=[mp1_ab_dn(1,2) mp1_ab_dn(1,1)];
Bab_dn=[0;1];
Cab_dn=[mp3_ab_dn(1,2) mp3_ab_dn(1,1)];

s1_ab_dn=ss(Aab_dn,Bab_dn,Cab_dn,Dab_dn);s2_ab_dn=c2d(s1_ab_dn,pTs);
Aab_d_dn=s2_ab_dn.a;
Bab_d_dn=s2_ab_dn.b;
Cab_d_dn=s2_ab_dn.c;


%regen brake
wn_rb=2.3*3;
n1_rb=[0 wn_rb^2];
d1_rb=[1 2*wn_rb wn_rb^2];
[Arb_,Brb_,Crb_,Drb_]=tf2ss(n1_rb,d1_rb);
mp1_rb=Arb_;mp2_rb=Brb_;mp3_rb=Crb_;mp4_rb=Drb_;
Arb_(1,:)=[0 1];Arb_(2,:)=[mp1_rb(1,2) mp1_rb(1,1)];
Brb_=[0;1];
Crb_=[mp3_rb(1,2) mp3_rb(1,1)];

s1_rb=ss(Arb_,Brb_,Crb_,Drb_);s2_rb=c2d(s1_rb,pTs);
Arb_d=s2_rb.a;
Brb_d=s2_rb.b;
Crb_d=s2_rb.c;

%regen brake when it comes down
wn_rb_dn=wn_rb*5;
n1_rb_dn=[0 wn_rb_dn^2];
d1_rb_dn=[1 2*wn_rb_dn wn_rb_dn^2];
[Arb_dn,Brb_dn,Crb_dn,Drb_dn]=tf2ss(n1_rb_dn,d1_rb_dn);
mp1_rb_dn=Arb_dn;mp2_rb_dn=Brb_dn;mp3_rb_dn=Crb_dn;mp4_rb_dn=Drb_dn;
Arb_dn(1,:)=[0 1];Arb_dn(2,:)=[mp1_rb_dn(1,2) mp1_rb_dn(1,1)];
Brb_dn=[0;1];
Crb_dn=[mp3_rb_dn(1,2) mp3_rb_dn(1,1)];

s1_rb_dn=ss(Arb_dn,Brb_dn,Crb_dn,Drb_dn);s2_rb_dn=c2d(s1_rb_dn,pTs);
Arb_d_dn=s2_rb_dn.a;
```

```
Brb_d_dn=s2_rb_dn.b;
Crb_d_dn=s2_rb_dn.c;


%traction
wn_tr=wn_rb; %same motor is used, assumed same transient
n1_tr=[0 wn_tr^2];
d1_tr=[1 2*wn_tr wn_tr^2];
[Atr_,Btr_,Ctr_,Dtr_]=tf2ss(n1_tr,d1_tr);
mp1_tr=Atr_;mp2_tr=Btr_;mp3_tr=Ctr_;mp4_tr=Dtr_;
Atr_(1,:)=[0 1];Atr_(2,:)=[mp1_tr(1,2) mp1_tr(1,1)];
Btr_=[0;1];
Ctr_=[mp3_tr(1,2) mp3_tr(1,1)];

s1_tr=ss(Atr_,Btr_,Ctr_,Dtr_);s2_tr=c2d(s1_tr,pTs);
Atr_d=s2_tr.a;
Btr_d=s2_tr.b;
Ctr_d=s2_tr.c;


%traction when it comes down
wn_tr_dn=wn_tr*5;
n1_tr_dn=[0 wn_tr_dn^2];
d1_tr_dn=[1 2*wn_tr_dn wn_tr_dn^2];
[Atr_dn,Btr_dn,Ctr_dn,Dtr_dn]=tf2ss(n1_tr_dn,d1_tr_dn);
mp1_tr_dn=Atr_dn;mp2_tr_dn=Btr_dn;mp3_tr_dn=Ctr_dn;mp4_tr_dn=Dtr_dn;
Atr_dn(1,:)=[0 1];Atr_dn(2,:)=[mp1_tr_dn(1,2) mp1_tr_dn(1,1)];
Btr_dn=[0;1];
Ctr_dn=[mp3_tr_dn(1,2) mp3_tr_dn(1,1)];

s1_tr_dn=ss(Atr_dn,Btr_dn,Ctr_dn,Dtr_dn);s2_tr_dn=c2d(s1_tr_dn,pTs);
Atr_d_dn=s2_tr_dn.a;
Btr_d_dn=s2_tr_dn.b;
Ctr_d_dn=s2_tr_dn.c;

back_step=[0;0;0;0;0;0];

integration_dummy=zeros(6,1);
integration_dummy_2=zeros(6,1);

i_x2e=zeros(6,1);
i_x2e_2=zeros(6,1);

u_ff=[0 0 0 0 0 0]';
u_ff_brake_temp=u_ff;
u_ff_rb_tmp=u_ff;
u_ff_rb=u_ff;
u_ff_ab=u_ff;
```

```
deadtime=ceil(t_delay/pTs);
cbinv_tr=inv(Ctr_d*Btr_d);
cbinv_tr_dn=inv(Ctr_d*Btr_d_dn);

cbinv_ab=inv(Cab_d*Bab_d);
cbinv_ab_dn=inv(Cab_d*Bab_d_dn);
cbinv_rb=inv(Crb_d*Brb_d);
cbinv_rb_dn=inv(Crb_d*Brb_d_dn);

catr=Ctr_d*Atr_d;
carb=Crb_d*Arb_d;
caab=Cab_d*Aab_d;
%feedforward

u_ff_calc=zeros(6,end_fifth_interval);
u_ff_rb_calc=u_ff_calc;u_ff_ab_calc=u_ff_calc;

for k=2:end_fifth_interval

%feedforward

for j=1:6

%         u_ff_calc(j,k)=(1/avg_force_ratio)*((x2r_ff(k)-x2r(k))/pTs)*mass_t(j)*pTs;
          u_ff_calc(j,k)=((x2r_ff(k)-x2r(k))/pTs)*mass_t(j)*pTs;

end %end of for for ff

end



      for k=2:end_fifth_interval
if k>end_third_interval&&velocity(1)<2.5
      break;
end

u_ff=u_ff_calc(:,k);

%         for k=1:5000
u_ff_data(k,:)=u_ff;
integration_interval=t(1:k);
for i=1:6
x1e(i)=x1r(k)-position(i);
x2e(i)=x2r(k)-velocity(i);
end

data_x1e(k,:)=x1e;
data_x2e(k,:)=x2e;
```

```
if k>1
for i=1:6
      integration_dummy(i)=pTs*(1/2)*(data_x2e(k,i)+data_x2e(k-1,i));
      i_x2e(i)=i_x2e(i)+integration_dummy(i);
end
end

data_i_x2e(:,k)=i_x2e;
if k>1
for i=1:6
      integration_dummy_2(i)=pTs*(1/2)*(data_i_x2e(i,k)+data_i_x2e(i,k));
      i_x2e_2(i)=i_x2e_2(i)+integration_dummy_2(i);
end
end
%control law for PI controller

p_gain=diag([1 1 1 1 1 1])*0.5;
i_gain=diag([1 1 1 1 1 1])*2;
i_gain_2=diag([1 1 1 1 1 1])*0;


b_gain=diag([1 1 1 1 1 1])*0.01*0; %antiwindup

i_gain=i_gain+diag(back_step);

u_cont=u_ff+p_gain*x2e+i_gain*i_x2e+i_gain_2*i_x2e_2; %position
u_before_sat=u_cont;
u_cont_data(k,:)=u_cont;
for i=1:6 %saturation and blending
if u_cont(i)>=0
            Force_m_max(i)=-
interp1(gTable_unt.trac_v,gTable_unt.trac_f,velocity(i),'linear');
else

Force_rb_max(i)=-
interp1(gTable_unt.reg_brake_v,gTable_unt.reg_brake_f,velocity(i),'linear');
%Mcar air
Force_ab_max(i)=-
interp1(gTable_unt.air_brake_v,gTable_unt.air_brake_f,velocity(i),'linear');
end
% first check total saturation
if u_cont(i)>=Force_m_max(i) %for traction
      u_cont(i)=Force_m_max(i);
else if u_cont(i)<(Force_ab_max(i)+Force_rb_max(i)) %for brakes
            u_cont(i)=Force_ab_max(i)+Force_rb_max(i);
      end
end
```

```
% brake blending
if u_cont(i)<=0
if u_cont(i)>Force_rb_max(i)
                Force_rb_des(i,k)=u_cont(i);
                Force_ab_des(i,k)=0;
        else
                Force_rb_des(i,k)=Force_rb_max(i);
                Force_ab_des(i,k)=u_cont(i)-Force_rb_des(i,k);
        end
end
end

%sanity
for i=1:6
    if u_cont(i)>120000
        u_cont(i)=120000;
    end
    if isnan(u_cont(i))

        u_cont(i)=data_u(k-1,i); %attention
    end
end
for i=1:6 % assign desired traction
if u_cont(i)>=0
   Force_tr_des(i,k)=u_cont(i);

end
end
%end of sanity
u_cont_after_sat=u_cont;
bst_sat=u_cont_after_sat-u_before_sat;
back_step=b_gain*bst_sat;
u_cont=u_cont+diag(back_step)*i_x2e;% antiwindup

u_cont_after_sat_data(k,:)=u_cont_after_sat;
data_u(k,:)=u_cont;
Force_m_max_data(k,:)=Force_m_max;
%running resistance equation is known, and feedforward force is assumed to
%already take consideration of running resistance
% dead time
for i=1:6
    if k<3 %initial condition

ab_count(i)=ab_count(i)+1;rb_count(i)=rb_count(i)+1;tr_count(i)=tr_count(i)+1;
    else
```

```
      %air brake
      if Force_ab_des(i,k-1)==0&&Force_ab(i,k-2)~=0
          x_ab((2*i-1):(2*i),k)=Aab_d_dn*x_ab((2*i-1):(2*i),k-
1)+Bab_d_dn*Force_ab_des(i,k-1);
      Force_ab(i,k)=Cab_d*x_ab((2*i-1):(2*i),k);
      ab_count(i)=0; % count increases until deadtime hits
    elseif                                  Force_ab_des(i,k-1)~=0&&Force_ab(i,k-
2)==0&&ab_count(i)<ceil(t_delay/pTs)
          x_ab((2*i-1):(2*i),k)=Aab_d*x_ab((2*i-1):(2*i),k-1); %dead time
      Force_ab(i,k)=Cab_d*x_ab((2*i-1):(2*i),k);
      ab_count(i)=ab_count(i)+1;
    elseif Force_ab_des(i,k-1)~=0&&Force_ab(i,k-2)~=0
          x_ab((2*i-1):(2*i),k)=Aab_d*x_ab((2*i-1):(2*i),k-
1)+Bab_d*Force_ab_des(i,k-1); %dead time
      Force_ab(i,k)=Cab_d*x_ab((2*i-1):(2*i),k);
      ab_count(i)=0;
    else
          x_ab((2*i-1):(2*i),k)=Aab_d_dn*x_ab((2*i-1):(2*i),k-
1)+Bab_d*Force_ab_des(i,k-1);
      Force_ab(i,k)=Cab_d*x_ab((2*i-1):(2*i),k);
      ab_count(i)=0;
    end

    %regen brake
    if Force_rb_des(i,k-1)==0&&Force_rb(i,k-2)~=0
          x_rb((2*i-1):(2*i),k)=Arb_d_dn*x_rb((2*i-1):(2*i),k-
1)+Brb_d_dn*Force_rb_des(i,k-1);
      Force_rb(i,k)=Crb_d*x_rb((2*i-1):(2*i),k);
      rb_count(i)=0; % count increases until deadtime hits
    elseif                                  Force_rb_des(i,k-1)~=0&&Force_rb(i,k-
2)==0&&rb_count(i)<ceil(t_delay/pTs)
          x_rb((2*i-1):(2*i),k)=Arb_d*x_rb((2*i-1):(2*i),k-1); %dead time
      Force_rb(i,k)=Crb_d*x_rb((2*i-1):(2*i),k);
      rb_count(i)=rb_count(i)+1;
    elseif Force_rb_des(i,k-1)~=0&&Force_rb(i,k-2)~=0
          x_rb((2*i-1):(2*i),k)=Arb_d*x_rb((2*i-1):(2*i),k-
1)+Brb_d*Force_rb_des(i,k-1); %dead time
      Force_rb(i,k)=Crb_d*x_rb((2*i-1):(2*i),k);
      rb_count(i)=0;
    else
          x_rb((2*i-1):(2*i),k)=Arb_d_dn*x_rb((2*i-1):(2*i),k-
1)+Bab_d*Force_rb_des(i,k-1);
      Force_rb(i,k)=Crb_d*x_rb((2*i-1):(2*i),k);
      rb_count(i)=0;
    end

      %traction
    if Force_tr_des(i,k-1)==0&&Force_tr(i,k-2)~=0
```

```
        x_tr((2*i-1):(2*i),k)=Atr_d_dn*x_tr((2*i-1):(2*i),k-
1)+Btr_d_dn*Force_tr_des(i,k-1);
      Force_tr(i,k)=Ctr_d*x_tr((2*i-1):(2*i),k);
      tr_count(i)=0; % count increases until deadtime hits
    elseif                                  Force_tr_des(i,k-1)~=0&&Force_tr(i,k-
2)==0&&tr_count(i)<ceil(t_delay/pTs)
        x_tr((2*i-1):(2*i),k)=Atr_d*x_tr((2*i-1):(2*i),k-1); %dead time
      Force_tr(i,k)=Ctr_d*x_tr((2*i-1):(2*i),k);
      tr_count(i)=tr_count(i)+1;
    elseif Force_tr_des(i,k-1)~=0&&Force_tr(i,k-2)~=0
        x_tr((2*i-1):(2*i),k)=Atr_d*x_tr((2*i-1):(2*i),k-1)+Btr_d*Force_tr_des(i,k-
1); %dead time
      Force_tr(i,k)=Ctr_d*x_tr((2*i-1):(2*i),k);
      tr_count(i)=0;
    else
        x_tr((2*i-1):(2*i),k)=Atr_d_dn*x_tr((2*i-1):(2*i),k-
1)+Bab_d*Force_tr_des(i,k-1);
      Force_tr(i,k)=Ctr_d*x_tr((2*i-1):(2*i),k);
      tr_count(i)=0;
    end

      end

end

%

Force_t(:,k)=Force_ab(:,k)+Force_rb(:,k)+Force_tr(:,k)+Force_d(:,k);          %add
force(conceptually)


for i=1:6
      position(i)=position_p(i)+velocity_p(i)*delta_t;
end
velocity(1)=velocity_p(1)+(       kk(1)*(position(2)-position(1))+cc(1)*(velocity(2)-
velocity(1))+Force_t(1,k)    )/mass_t(1)*delta_t;
for i=2:5
      velocity(i)=velocity_p(i)+(                  kk(i-1)*(position(i-1)-position(i))+
kk(i)*(position(i+1)-position(i))+cc(i-1)*(velocity(i-1)-
velocity(i))+cc(i)*(velocity(i+1)-velocity(i))+Force_t(i,k) )/mass_t(i)*delta_t;
end

velocity(6)=velocity_p(6)+(       kk(5)*(position(5)-position(6))+cc(5)*(velocity(5)-
velocity(6))+Force_t(6,k)    )/mass_t(6)*delta_t;

for i=1:6
      if velocity(i)<0
            velocity(i)=0;
```

```
            position(i)=position_p(i);
        else if velocity(i)>80/3.6
                velocity(i)=80/3.6;
            end
        end
end


data_position(k,:)=position;
data_velocity(k,:)=velocity;


% at the end of loop
position_p=position;
velocity_p=velocity;
count=count+1;
% end of control loop
        end

  %rhc takes over

s=tf('s');
Ts=0.1; %sampling time of 0.1 seconds for controller
pTs=0.001; %sampling time of discretized plant for testing
horizon=20; %should cover at least 2 seconds window
fault_flag=0; % determines which car has total failure
t_delay=0.2*(1+delay_dev); %model assumed pure delay of 0.2 second
delay_constant=ceil(t_delay/Ts);
freq_within_calc=floor(Ts/pTs);
wn=2.3*(1+wn_dev);
n1=[0 wn^2];
d1=[1 2*wn wn^2];
tf1=tf(n1,d1);
tf1=tf1*exp(-t_delay*s);
[n1,d1]=tfdata(tf1,'v');
[Aab_r,Bab_r,Cab_r,Dab_r]=tf2ss(n1,d1);
mp1=Aab_r;mp2=Bab_r;mp3=Cab_r;mp4=Dab_r;
Aab_r(1,:)=[0 1];Aab_r(2,:)=[mp1(1,2) mp1(1,1)];
Bab_r=[0;1];
Cab_r=[mp3(1,2) mp3(1,1)];


% Air brake CCF
Aab=[0 1 0;0 0 1;-52.9 -51.29 -14.6];
Bab=[0;0;1];
Cab=[52.9 -5.29 0];

m1=42153+6000; %mc1,    unit: kg
```

```
m2=40463+3000; %m
m3=40463+5500; %m
m4=40463+2000; %m
m5=40463+4000; %m
m6=41990+3500; %mc2

%mass deviation. 0.1 means real mass is +10 percent of real value
dev_m_1=dev_m;
dev_m_2=dev_m;
dev_m_3=dev_m;
dev_m_4=dev_m;
dev_m_5=dev_m;
dev_m_6=dev_m;

m1r=m1*(1+dev_m_1); % mass error for real car
m2r=m2*(1+dev_m_2); % mass error for real car
m3r=m3*(1+dev_m_3); % mass error for real car
m4r=m4*(1+dev_m_4); % mass error for real car
m5r=m5*(1+dev_m_5); % mass error for real car
m6r=m6*(1+dev_m_6); % mass error for real car


kk=[3.6 3.6 3.6 3.6 3.6]*10^7;
cc=[8333    8333    8333    8333    8333];

k1=kk(1);
k2=kk(2);
k3=kk(3);
k4=kk(4);
k5=kk(5);


c1=cc(1);
c2=cc(2);
c3=cc(3);
c4=cc(4);
c5=cc(5);


   A_real = [ 0    1              zeros(1,22);
      -k1/m1r       -c1/m1r          Cab_r/m1r     k1/m1r          c1/m1r    zeros(1,18);
      zeros(2,2)                    Aab_r          zeros(2,20);
        0                    0                 zeros(1,2)          0                    1
zeros(1,18);
       k1/m2r          c1/m2r       zeros(1,2)  (-k1-k2)/m2r    (-c1-c2)/m2r    Cab_r/m2r
k2/m2r          c2/m2r          zeros(1,14);
       zeros(2,6)                                                                  Aab_r
zeros(2,16);
```

```
        0                0              zeros(1,2)      0                0              zeros(1,2)
0               1              zeros(1,14);
        0                0              zeros(1,2)      k2/m3r    c2/m3r         zeros(1,2)  (-k2-
k3)/m3r    (-c2-c3)/m3r Cab_r/m3r    k3/m3r            c3/m3r         zeros(1,10);
      zeros(2,10)
Aab_r zeros(2,12);
        0                0              zeros(1,2)      0                0              zeros(1,2)      0
0            zeros(1,2)      0                1              zeros(1,10);
        0                0              zeros(1,2)      0                0              zeros(1,2)
k3/m4r            c3/m4r            zeros(1,2)  (-k3-k4)/m4r    (-c3-c4)/m4r  Cab_r/m4r
k4/m4r            c4/m4r zeros(1,6);
      zeros(2,14)
Aab_r zeros(2,8);
      zeros(1,16)                                                                    0                1
zeros(1,6);
      zeros(1,12)   k4/m5r          c4/m5r          zeros(1,2)  (-k4-k5)/m5r    (-c4-c5)/m5r
Cab_r/m5r    k5/m5r            c5/m5r zeros(1,2);
      zeros(2,18)
Aab_r zeros(2,4);
      zeros(1,20)                                                                  0                1
zeros(1,2);
      zeros(1,16)   k5/m6r                c5/m6r          zeros(1,2)  -k5/m6r          -c5/m6r
Cab_r/m6r
      zeros(2,22)                                                                                  Aab_r
      ];


temp_b=[0 0 0 1]';

B_real=[ temp_b zeros(4,5);
              zeros(4,1) temp_b zeros(4,4);
              zeros(4,2) temp_b zeros(4,3);
              zeros(4,3) temp_b zeros(4,2);
              zeros(4,4) temp_b zeros(4,1);
              zeros(4,5) temp_b];



C_real=[0 1 zeros(1,22);
    zeros(1,4) 0 1 zeros(1,18);
    zeros(1,8) 0 1 zeros(1,14);
    zeros(1,12) 0 1 zeros(1,10);
    zeros(1,16) 0 1 zeros(1,6);
    zeros(1,20) 0 1 zeros(1,2)];

D_real=zeros(6,6);

Adn=zeros(2,2,6);
```

```
plant_model_A=zeros(24,24,64); %plant models will be stared to accomodate each
brking situation
plant_model_B=zeros(24,6,64);

for car_1=0:1
    for car_2=0:1
        for car_3=0:1
            for car_4=0:1
                for car_5=0:1
                    for car_6=0:1
                        if car_1==0
      Adn(:,:,1)=Aab_dn;
else Adn(:,:,1)=Aab_r;
end

if car_2==0
      Adn(:,:,2)=Aab_dn;
else Adn(:,:,2)=Aab_r;
end

if car_3==0
      Adn(:,:,3)=Aab_dn;
else Adn(:,:,3)=Aab_r;
end

if car_4==0
      Adn(:,:,4)=Aab_dn;
else Adn(:,:,4)=Aab_r;
end

if car_5==0
      Adn(:,:,5)=Aab_dn;
else Adn(:,:,5)=Aab_r;
end

if car_6==0
      Adn(:,:,6)=Aab_dn;
else Adn(:,:,6)=Aab_r;
end
temp_index=car_1*2^0+car_2*2^1+car_3*2^2+car_4*2^3+car_5*2^4+car_6*2^5+
1;

    %calculation of plant model with corresponding braking situation
  A_real_dn = [ 0    1              zeros(1,22);
     -k1/m1r      -c1/m1r          Cab_r/m1r     k1/m1r           c1/m1r    zeros(1,18);
     zeros(2,2)                   Adn(:,:,1)          zeros(2,20);
       0                    0                 zeros(1,2)            0                    1
zeros(1,18);
```

```
      k1/m2r          c1/m2r       zeros(1,2)  (-k1-k2)/m2r   (-c1-c2)/m2r   Cab_r/m2r
k2/m2r          c2/m2r          zeros(1,14);
      zeros(2,6)                                                                  Adn(:,:,2)
zeros(2,16);
        0               0           zeros(1,2)      0               0           zeros(1,2)
0               1           zeros(1,14);
        0               0           zeros(1,2)    k2/m3r    c2/m3r      zeros(1,2)  (-k2-
k3)/m3r   (-c2-c3)/m3r Cab_/m3r   k3/m3r          c3/m3r       zeros(1,10);
      zeros(2,10)
Adn(:,:,3) zeros(2,12);
        0               0           zeros(1,2)      0               0       zeros(1,2)      0
0           zeros(1,2)      0               1       zeros(1,10);
        0               0           zeros(1,2)      0               0       zeros(1,2)
k3/m4r          c3/m4r          zeros(1,2)  (-k3-k4)/m4r   (-c3-c4)/m4r   Cab_r/m4r
k4/m4r          c4/m4r zeros(1,6);
      zeros(2,14)
Adn(:,:,4) zeros(2,8);
      zeros(1,16)                                                                 0               1
zeros(1,6);
      zeros(1,12)   k4/m5r          c4/m5r      zeros(1,2)  (-k4-k5)/m5r   (-c4-c5)/m5r
Cab_r/m5r   k5/m5r          c5/m5r zeros(1,2);
      zeros(2,18)
Adn(:,:,5) zeros(2,4);
      zeros(1,20)                                                                 0               1
zeros(1,2);
      zeros(1,16)   k5/m6r              c5/m6r      zeros(1,2)      -k5/m6r         -c5/m6r
Cab_r/m6r
      zeros(2,22)
Adn(:,:,6)
      ];


temp_b=[0 0 0 1]';

B_real_dn=[ temp_b zeros(4,5);
            zeros(4,1) temp_b zeros(4,4);
            zeros(4,2) temp_b zeros(4,3);
            zeros(4,3) temp_b zeros(4,2);
            zeros(4,4) temp_b zeros(4,1);
            zeros(4,5) temp_b];



C_real_dn=[0 1 zeros(1,22);
    zeros(1,4) 0 1 zeros(1,18);
    zeros(1,8) 0 1 zeros(1,14);
    zeros(1,12) 0 1 zeros(1,10);
    zeros(1,16) 0 1 zeros(1,6);
```

```
    zeros(1,20) 0 1 zeros(1,2)];

D_real_dn=zeros(6,6);


s5_r=ss(A_real_dn,B_real_dn,C_real_dn,D_real_dn);s6_r=c2d(s5_r,pTs);  %discretiz
e by plant sampling time, which is different from the rhc caculation frequency
Ad_real_dn=s6_r.a;
Bd_real_dn=s6_r.b;


plant_model_A(:,:,temp_index)=Ad_real_dn;
plant_model_B(:,:,temp_index)=Bd_real_dn;

                        end
                    end
                end
            end
    end
end

ab_flag=zeros(1,6);

%point mass model
Ao1=[0 1;0 0];
Bo1=[0;1/m1];

Ao2=[0 1;0 0];
Bo2=[0;1/m2];

Ao3=[0 1;0 0];
Bo3=[0;1/m3];

Ao4=[0 1;0 0];
Bo4=[0;1/m4];

Ao5=[0 1;0 0];
Bo5=[0;1/m5];

Ao6=[0 1;0 0];
Bo6=[0;1/m6];

%make model for each car with same air brake

B1=[0;0;0;0;1];
A1=[Ao1 Bo1*Cab;zeros(3,2) Aab];
C1=[0 1 0 0 0];
```

```
B2=[0;0;0;0;1];
A2=[Ao2 Bo2*Cab;zeros(3,2) Aab];
C2=[0 1 0 0 0];

B3=[0;0;0;0;1];
A3=[Ao3 Bo3*Cab;zeros(3,2) Aab];
C3=[0 1 0 0 0];

B4=[0;0;0;0;1];
A4=[Ao4 Bo4*Cab;zeros(3,2) Aab];
C4=[0 1 0 0 0];

B5=[0;0;0;0;1];
A5=[Ao5 Bo5*Cab;zeros(3,2) Aab];
C5=[0 1 0 0 0];

B6=[0;0;0;0;1];
A6=[Ao6 Bo6*Cab;zeros(3,2) Aab];
C6=[0 1 0 0 0];

%%discretization

s1=ss(A1,B1,C1,[0]);s2=c2d(s1,Ts);
Ad1=s2.a;
Bd1=s2.b;

s1=ss(A2,B2,C2,[0]);s2=c2d(s1,Ts);
Ad2=s2.a;
Bd2=s2.b;

s1=ss(A3,B3,C3,[0]);s2=c2d(s1,Ts);
Ad3=s2.a;
Bd3=s2.b;

s1=ss(A4,B4,C4,[0]);s2=c2d(s1,Ts);
Ad4=s2.a;
Bd4=s2.b;

s1=ss(A5,B5,C5,[0]);s2=c2d(s1,Ts);
Ad5=s2.a;
Bd5=s2.b;

s1=ss(A6,B6,C6,[0]);s2=c2d(s1,Ts);
Ad6=s2.a;
Bd6=s2.b;
```

```
s3=ss(A_real,B_real,C_real,D_real);s4=c2d(s3,pTs); %discretize by plant sampling time, which is different from the rhc caculation frequency
Ad_real=s4.a;
Bd_real=s4.b;
Cd_real=s4.c;
Dd_real=s4.d;

[row, dima]=size(A1);
dimb=1;


%%%%% make equality condition for each car

temp=[Ad1 -eye(dima)];

   Aeq1_top=[eye(dima) zeros(dima,horizon+horizon*dima)];
   Aeq1=Aeq1_top;
for i=1:horizon

      Aeq1_tmp1=[zeros(dima,dima*(i-1)) temp zeros(dima, (horizon-1)*dima-(dima-dimb)*(i-1)) Bd1 zeros(dima, horizon-i)];
      Aeq1=[Aeq1; Aeq1_tmp1];
end


n_constraints=1;
r1=[1 0 0 0 0];
Ain1_top=[r1 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain1_top_tmp=[zeros(n_constraints, dima*i) r1 zeros(n_constraints, horizon*(dima+1)-dima*i)];
      Ain1_top=[Ain1_top;Ain1_top_tmp];
end

r2=[0 1 0 0 0];
Ain1_top_2=[r2 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain1_top_tmp_2=[zeros(n_constraints, dima*i) r2 zeros(n_constraints, horizon*(dima+1)-dima*i)];
      Ain1_top_2=[Ain1_top_2;Ain1_top_tmp_2];
end


Ain1_bottom=[zeros(horizon, (horizon+1)*dima) eye(horizon)];

Ain1=[Ain1_top;-Ain1_top;Ain1_top_2;-Ain1_top_2;Ain1_bottom;-Ain1_bottom];
```

```
% 2


temp=[Ad2 -eye(dima)];

   Aeq2_top=[eye(dima) zeros(dima,horizon+horizon*dima)];
   Aeq2=Aeq2_top;
for i=1:horizon

      Aeq2_tmp1=[zeros(dima,dima*(i-1)) temp zeros(dima, (horizon-1)*dima-
(dima-dimb)*(i-1)) Bd2 zeros(dima, horizon-i)];
      Aeq2=[Aeq2; Aeq2_tmp1];
end


n_constraints=1;
r1=[1 0 0 0 0];
Ain2_top=[r1 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain2_top_tmp=[zeros(n_constraints, dima*i) r1 zeros(n_constraints,
horizon*(dima+1)-dima*i)];
      Ain2_top=[Ain2_top;Ain2_top_tmp];
end

r2=[0 1 0 0 0];
Ain2_top_2=[r2 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain2_top_tmp_2=[zeros(n_constraints, dima*i) r2 zeros(n_constraints,
horizon*(dima+1)-dima*i)];
      Ain2_top_2=[Ain2_top_2;Ain2_top_tmp_2];
end


Ain2_bottom=[zeros(horizon, (horizon+1)*dima) eye(horizon)];

Ain2=[Ain2_top;-Ain2_top;Ain2_top_2;-Ain2_top_2;Ain2_bottom;-Ain2_bottom];

%3

temp=[Ad3 -eye(dima)];

   Aeq3_top=[eye(dima) zeros(dima,horizon+horizon*dima)];
   Aeq3=Aeq3_top;
for i=1:horizon

      Aeq3_tmp1=[zeros(dima,dima*(i-1)) temp zeros(dima, (horizon-1)*dima-
(dima-dimb)*(i-1)) Bd3 zeros(dima, horizon-i)];
      Aeq3=[Aeq3; Aeq3_tmp1];
```

```
end


n_constraints=1;
r1=[1 0 0 0 0];
Ain3_top=[r1 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain3_top_tmp=[zeros(n_constraints,  dima*i)  r1  zeros(n_constraints,
horizon*(dima+1)-dima*i)];
      Ain3_top=[Ain3_top;Ain3_top_tmp];
end

r2=[0 1 0 0 0];
Ain3_top_2=[r2 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain3_top_tmp_2=[zeros(n_constraints,  dima*i)  r2  zeros(n_constraints,
horizon*(dima+1)-dima*i)];
      Ain3_top_2=[Ain3_top_2;Ain3_top_tmp_2];
end


Ain3_bottom=[zeros(horizon, (horizon+1)*dima) eye(horizon)];

Ain3=[Ain3_top;-Ain3_top;Ain3_top_2;-Ain3_top_2;Ain3_bottom;-Ain3_bottom];

% 4
temp=[Ad4 -eye(dima)];

   Aeq4_top=[eye(dima) zeros(dima,horizon+horizon*dima)];
   Aeq4=Aeq4_top;
for i=1:horizon

      Aeq4_tmp1=[zeros(dima,dima*(i-1))  temp  zeros(dima,  (horizon-1)*dima-
(dima-dimb)*(i-1)) Bd4 zeros(dima, horizon-i)];
      Aeq4=[Aeq4; Aeq4_tmp1];
end


n_constraints=1;
r1=[1 0 0 0 0];
Ain4_top=[r1 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain4_top_tmp=[zeros(n_constraints,  dima*i)  r1  zeros(n_constraints,
horizon*(dima+1)-dima*i)];
      Ain4_top=[Ain4_top;Ain4_top_tmp];
end

r2=[0 1 0 0 0];
```

```
Ain4_top_2=[r2 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
    Ain4_top_tmp_2=[zeros(n_constraints, dima*i) r2 zeros(n_constraints, horizon*(dima+1)-dima*i)];
    Ain4_top_2=[Ain4_top_2;Ain4_top_tmp_2];
end


Ain4_bottom=[zeros(horizon, (horizon+1)*dima) eye(horizon)];

Ain4=[Ain4_top;-Ain4_top;Ain4_top_2;-Ain4_top_2;Ain4_bottom;-Ain4_bottom];

%5

temp=[Ad5 -eye(dima)];

  Aeq5_top=[eye(dima) zeros(dima,horizon+horizon*dima)];
  Aeq5=Aeq5_top;
for i=1:horizon

    Aeq5_tmp1=[zeros(dima,dima*(i-1)) temp zeros(dima, (horizon-1)*dima-(dima-dimb)*(i-1)) Bd5 zeros(dima, horizon-i)];
    Aeq5=[Aeq5; Aeq5_tmp1];
end


n_constraints=1;
r1=[1 0 0 0 0];
Ain5_top=[r1 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
    Ain5_top_tmp=[zeros(n_constraints, dima*i) r1 zeros(n_constraints, horizon*(dima+1)-dima*i)];
    Ain5_top=[Ain5_top;Ain5_top_tmp];
end

r2=[0 1 0 0 0];
Ain5_top_2=[r2 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
    Ain5_top_tmp_2=[zeros(n_constraints, dima*i) r2 zeros(n_constraints, horizon*(dima+1)-dima*i)];
    Ain5_top_2=[Ain5_top_2;Ain5_top_tmp_2];
end


Ain5_bottom=[zeros(horizon, (horizon+1)*dima) eye(horizon)];

Ain5=[Ain5_top;-Ain5_top;Ain5_top_2;-Ain5_top_2;Ain5_bottom;-Ain5_bottom];
```

```
%6

temp=[Ad6 -eye(dima)];

   Aeq6_top=[eye(dima) zeros(dima,horizon+horizon*dima)];
   Aeq6=Aeq6_top;
for i=1:horizon

      Aeq6_tmp1=[zeros(dima,dima*(i-1)) temp zeros(dima, (horizon-1)*dima-(dima-dimb)*(i-1)) Bd6 zeros(dima, horizon-i)];
      Aeq6=[Aeq6; Aeq6_tmp1];
end


n_constraints=1;
r1=[1 0 0 0 0];
Ain6_top=[r1 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain6_top_tmp=[zeros(n_constraints, dima*i) r1 zeros(n_constraints, horizon*(dima+1)-dima*i)];
      Ain6_top=[Ain6_top;Ain6_top_tmp];
end

r2=[0 1 0 0 0];
Ain6_top_2=[r2 zeros(n_constraints, horizon*(dima+1))];
for i=1:horizon
      Ain6_top_tmp_2=[zeros(n_constraints, dima*i) r2 zeros(n_constraints, horizon*(dima+1)-dima*i)];
      Ain6_top_2=[Ain6_top_2;Ain6_top_tmp_2];
end


Ain6_bottom=[zeros(horizon, (horizon+1)*dima) eye(horizon)];

Ain6=[Ain6_top;-Ain6_top;Ain6_top_2;-Ain6_top_2;Ain6_bottom;-Ain6_bottom];

alpha=20;beta=0;gamma=0;delta=23000;

%%end of constraint setting
initial_position_error=0;
ref_v_at_starting_pt=velocity(1);
brake_force_ratio=0.45; %lower value would stop slower but show more robustness to variation.
max_mass=50000; % assumed maximum mass of 1 car with passengers
min_force=23000; % assumed max force of 1 air brake
a=-brake_force_ratio*min_force/max_mass;
% stop_t=ref_v_at_starting_pt*(-1/a);
distance_to_stop=station_d-position(1);
```

```
stop_t=distance_to_stop/velocity(1)*2;

Bin=[ones(horizon+1,n_constraints)*(distance_to_stop+0.001);
ones(horizon+1,n_constraints)*0.01;                ones(horizon+1,n_constraints)*alpha;
ones(horizon+1,n_constraints)*beta;  ones(horizon,1)*gamma;ones(horizon,1)*delta];


a=-velocity(1)/stop_t;
seconds=stop_t+3;
ttt=0:Ts:seconds+3;
distance_left=station_d-position(1);
stop_t=(3*distance_left/velocity(1));
lin_slope=velocity(1)/stop_t^2;

snd_ord_int=0:Ts:stop_t;

q_v=lin_slope*(snd_ord_int-stop_t).^2;
q_p=lin_slope*((1/3)*snd_ord_int.^3-stop_t*snd_ord_int.^2+stop_t^2*snd_ord_int);

x2rhc=zeros(1,length(ttt));
x1rhc=distance_left*ones(1,length(ttt));
x2rhc(1:length(q_v))=q_v;
x1rhc(1:length(q_p))=q_p;
n_interval=ceil(seconds/Ts);
weight=[1 1 0 0 0];
H=zeros(1,horizon);
for i=1:horizon+1

H=[weight H];
end
H=diag(H);


H1=H;
H2=H;
H3=H;
H4=H;
H5=H;
H6=H;


init=[0 velocity(1) 0 0 0]'; %initialization
theta=[init; zeros(horizon*(dima+1),n_constraints)];
theta_r=[init(1:4); zeros(horizon*(dima),n_constraints)]; %careful when adjusting


theta1=theta;
theta2=theta;
```

```
theta3=theta;
theta4=theta;
theta5=theta;
theta6=theta;

theta1_r=theta_r;
theta2_r=theta_r;
theta3_r=theta_r;
theta4_r=theta_r;
theta5_r=theta_r;
theta6_r=theta_r;



x_next=[theta1(1:dima);theta2(1:dima);theta3(1:dima);theta4(1:dima);theta5(1:dima)
;theta6(1:dima)];
x(1,1:30)=x_next;
x_next_r=[theta1_r(1:4);theta2_r(1:4);theta3_r(1:4);theta4_r(1:4);theta5_r(1:4);theta
6_r(1:4)]; %carefully check this when modifying
x_r(1,1:24)=x_next_r;
disturbance(1,1:6)=[0 0 0 0 0 0];
%start of optimization loop

Force_ab_des=zeros(6,n_interval);
Force_ab=zeros(6,freq_within_calc);
u1r=zeros(1,freq_within_calc);
u2r=u1r;u3r=u1r;u4r=u1r;u5r=u1r;u6r=u1r;


for i=1:n_interval;



    f=[];
     for j=0:horizon
           f=[f x1rhc(i+j) x2rhc(i+j) zeros(1,dima-2)];
     end
     f=-[f zeros(1,horizon)]';

     %for car 1
     Beq1=[theta1(1:dima); zeros(dima*horizon,1)];
     if i<delay_constant+1||fault_flag==1
           u1(i)=0;
     else
     theta_opt1=quadprog(H,f,Ain1,Bin, Aeq1, Beq1);
     u1(i)=theta_opt1(dima*(horizon+1)+1);
     end
    f=[];
```

```
 for j=0:horizon
       f=[f x1rhc(i+j) x2rhc(i+j) zeros(1,dima-2)];
 end
 f=-[f zeros(1,horizon)]';


 Beq2=[theta2(1:dima); zeros(dima*horizon,1)];
 if i<delay_constant+1||fault_flag==2
       u2(i)=0;
 else
 theta_opt2=quadprog(H,f,Ain2,Bin, Aeq2, Beq2);
 u2(i)=theta_opt2(dima*(horizon+1)+1);
 end
f=[];
 for j=0:horizon
       f=[f x1rhc(i+j) x2rhc(i+j) zeros(1,dima-2)];
 end
 f=-[f zeros(1,horizon)]';
 Beq3=[theta3(1:dima); zeros(dima*horizon,1)];
 if i<delay_constant+1||fault_flag==3
       u3(i)=0;
 else
 theta_opt3=quadprog(H,f,Ain3,Bin, Aeq3, Beq3);
 u3(i)=theta_opt3(dima*(horizon+1)+1);
 end
f=[];
 for j=0:horizon
       f=[f x1rhc(i+j) x2rhc(i+j) zeros(1,dima-2)];
 end
 f=-[f zeros(1,horizon)]';

 Beq4=[theta4(1:dima); zeros(dima*horizon,1)];
       if i<delay_constant+1||fault_flag==4
       u4(i)=0;
 else
 theta_opt4=quadprog(H,f,Ain4,Bin, Aeq4, Beq4);
 u4(i)=theta_opt4(dima*(horizon+1)+1)*sign(x_next(17));
       end
f=[];
 for j=0:horizon
       f=[f x1rhc(i+j) x2rhc(i+j) zeros(1,dima-2)];
 end
 f=-[f zeros(1,horizon)]';
 Beq5=[theta5(1:dima); zeros(dima*horizon,1)];
 if i<delay_constant+1||fault_flag==5
       u5(i)=0;
 else
 theta_opt5=quadprog(H,f,Ain5,Bin, Aeq5, Beq5);    %normal
```

```
      u5(i)=theta_opt5(dima*(horizon+1)+1);

      end
     f=[];
      for j=0:horizon
            f=[f x1rhc(i+j) x2rhc(i+j) zeros(1,dima-2)];
      end
      f=-[f zeros(1,horizon)]';
      Beq6=[theta6(1:dima); zeros(dima*horizon,1)];
      if i<delay_constant+1||fault_flag==6
            u6(i)=0;
      else
      theta_opt6=quadprog(H,f,Ain6,Bin, Aeq6, Beq6);
      u6(i)=theta_opt6(dima*(horizon+1)+1);
      end
      if i==1

temp_x_next_r=[theta1_r(1:4);theta2_r(1:4);theta3_r(1:4);theta4_r(1:4);theta5_r(1:4)
;theta6_r(1:4)]; % initialize
      end
      %inside the loop

Force_ab_des(1,i)=u1(i);Force_ab_des(2,i)=u2(i);Force_ab_des(3,i)=u3(i);Force_ab
_des(4,i)=u4(i);Force_ab_des(5,i)=u5(i);Force_ab_des(6,i)=u6(i);

%update more finer frequency for the plant model default 100hz
                        temp_next_r=x_next_r;

            for ddd=1:(freq_within_calc)
                        for ii=1:6
      if ddd<2 %initial condition
                  %air brake
      for ii=1:6
      %air brake
      if Force_ab_des(ii,i)==0&&Force_ab(ii,freq_within_calc)~=0 %when brake is
released
      Force_ab(ii,ddd)=0;
      ab_flag(ii)=0; % 0 when released
      ab_count(ii)=0; % count increases until deadtime hits
     elseif
Force_ab_des(ii,i)~=0&&Force_ab(ii,freq_within_calc)==0&&ab_count(ii)<ceil(t_d
elay/pTs) %when resting brake is called for action
     Force_ab(ii,ddd)=0;
      ab_flag(ii)=0;
      ab_count(ii)=ab_count(ii)+1;% count increases until deadtime hits
        elseif
Force_ab_des(ii,i)~=0&&Force_ab(ii,freq_within_calc)==0&&ab_count(ii)>=ceil(t_
delay/pTs)% when count is full
```

```
      Force_ab(ii,ddd)=Force_ab_des(ii,i);
      ab_flag(ii)=1;
      ab_count(ii)=0;
    elseif  Force_ab_des(ii,i)~=0&&Force_ab(ii,freq_within_calc)~=0  %  when
working brake just changes force it exerts
      Force_ab(ii,ddd)=Force_ab_des(ii,i);
      ab_flag(ii)=1;
      ab_count(ii)=0;
      else % not active
            ab_flag(ii)=0;
      end %end of airbrake
      end % end of ii for loop


      else
      for ii=1:6
      %air brake
      if Force_ab_des(ii,i)==0&&Force_ab(ii,ddd-1)~=0 %when brake is released
      Force_ab(ii,ddd)=0;
      ab_flag(ii)=0; % 0 when released
      ab_count(ii)=0; % count increases until deadtime hits
    elseif                                  Force_ab_des(ii,i)~=0&&Force_ab(ii,ddd-
1)==0&&ab_count(ii)<ceil(t_delay/pTs) %when resting brake is called for action
    Force_ab(ii,ddd)=0;
      ab_flag(ii)=0;
      ab_count(ii)=ab_count(ii)+1;% count increases until deadtime hits
       elseif                                 Force_ab_des(ii,i)~=0&&Force_ab(ii,ddd-
1)==0&&ab_count(ii)>=ceil(t_delay/pTs)% when count is full
      Force_ab(ii,ddd)=Force_ab_des(ii,i);
      ab_flag(ii)=1;
      ab_count(ii)=0;
    elseif Force_ab_des(ii,i)~=0&&Force_ab(ii,ddd-1)~=0 % when working brake
just changes force it exerts
      Force_ab(ii,ddd)=Force_ab_des(ii,i);
      ab_flag(ii)=1;
      ab_count(ii)=0;
      else % not active
%               Force_ab(ii,i)=0;
            ab_flag(ii)=0;
              % ab_count(ii)=0;
      end %end of airbrake
      end % end of ii for loop
      end
                        end% end of ddd for loop

u1r(ddd)=Force_ab(1,ddd);
u2r(ddd)=Force_ab(2,ddd);
u3r(ddd)=Force_ab(3,ddd);
```

```
u4r(ddd)=Force_ab(4,ddd);
u5r(ddd)=Force_ab(5,ddd);
u6r(ddd)=Force_ab(6,ddd);

plant_index=ab_flag(1)*2^0+ab_flag(2)*2^1+ab_flag(3)*2^2+ab_flag(4)*2^3+ab_fl
ag(5)*2^4+ab_flag(6)*2^5+1;

      x_next_r=plant_model_A(:,:,plant_index)*temp_next_r
+plant_model_B(:,:,plant_index)*[u1r(ddd);u2r(ddd);u3r(ddd);u4r(ddd);u5r(ddd);u6r
(ddd)]; % time delay of 0.2sec
       temp_next_r=x_next_r;
           end



      if x_next_r(2)<0
            x_next_r(2)=0;
            x_next_r(1)=theta1_r(1);
      end

      if x_next_r(6)<0
            x_next_r(6)=0;
            x_next_r(5)=theta2_r(1);
      end

      if x_next_r(10)<0
            x_next_r(10)=0;
            x_next_r(9)=theta3_r(1);
      end

      if x_next_r(14)<0
            x_next_r(14)=0;
            x_next_r(13)=theta4_r(1);
      end

      if x_next_r(18)<0
            x_next_r(18)=0;
            x_next_r(17)=theta5_r(1);
      end

      if x_next_r(22)<0
            x_next_r(22)=0;
            x_next_r(21)=theta6_r(1);
      end

      % update theta
      theta1_r(1:4)=x_next_r(1:4);
      theta2_r(1:4)=x_next_r(5:8);
```

```
    theta3_r(1:4)=x_next_r(9:12);
    theta4_r(1:4)=x_next_r(13:16);
    theta5_r(1:4)=x_next_r(17:20);
    theta6_r(1:4)=x_next_r(21:24);

    theta1(1:dima)=Ad1*[x_next_r(1:2);theta1(3:5)]+Bd1*u1(i);
    theta2(1:dima)=Ad2*[x_next_r(5:6);theta2(3:5)]+Bd2*u2(i);
    theta3(1:dima)=Ad3*[x_next_r(9:10);theta3(3:5)]+Bd3*u3(i);
    theta4(1:dima)=Ad4*[x_next_r(13:14);theta4(3:5)]+Bd4*u4(i);
    theta5(1:dima)=Ad5*[x_next_r(17:18);theta5(3:5)]+Bd5*u5(i);
    theta6(1:dima)=Ad6*[x_next_r(21:22);theta6(3:5)]+Bd6*u6(i);


    x_r(i+1,:)=x_next_r';
end



Figure(6)
hold on;
plot(ttt(1:n_interval),x1rhc(1:n_interval)-x_r(1:n_interval,1)','Color',[1                0
0],'LineWidth',2);xlabel('time(seconds)');ylabel('position error(meters)');
plot(ttt(1:n_interval),x1rhc(1:n_interval)-x_r(1:n_interval,5)','Color',[1 1 0]);
plot(ttt(1:n_interval),x1rhc(1:n_interval)-x_r(1:n_interval,9)','Color',[1 0 1]);
plot(ttt(1:n_interval),x1rhc(1:n_interval)-x_r(1:n_interval,13)','Color',[0 1 0]);
plot(ttt(1:n_interval),x1rhc(1:n_interval)-x_r(1:n_interval,17)','Color',[0 1 1]);
plot(ttt(1:n_interval),x1rhc(1:n_interval)-x_r(1:n_interval,21)','Color',[0 0 1]);
hold off;
legend('Car 1','Car 2','Car 3','Car 4','Car 5','Car 6');
set(gca,'FontSize', 20);
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Position error(m)','fontsize',20);


%combine 2DOF interval data and LRHC data

%reference
mul_v=ceil(Ts/pTs);
for i_20=1:length(x2rhc)

    tmp_combined_1((i_20-1)*mul_v+1:(i_20-
1)*mul_v+mul_v)=x1rhc(i_20)+position(1);
    tmp_combined_2((i_20-1)*mul_v+1:(i_20-1)*mul_v+mul_v)=x2rhc(i_20);
end
```

```
x1r_combined=[x1r(1:k) tmp_combined_1];
x2r_combined=[x2r(1:k) tmp_combined_2];

time_scale_1=0:pTs:length(x2r)*pTs-pTs;
time_scale_2=0:pTs:length(x2r_combined)*pTs-pTs;
for i_20=1:n_interval

      tmp_combined_pos((i_20-1)*mul_v+1:(i_20-
1)*mul_v+mul_v)=x_r(i_20,1)+position(1);
      tmp_combined_vel((i_20-1)*mul_v+1:(i_20-1)*mul_v+mul_v)=x_r(i_20,2);
end
position_combined=[data_position(:,1)' tmp_combined_pos];
velocity_combined=[data_velocity(:,1)' tmp_combined_vel];

time_scale_3=0:pTs:length(position_combined)*pTs-pTs;

Figure(8)
hold  on;plot(time_scale_3,x2r_combined(1:length(position_combined)),'Color',[1  0
1]);plot(time_scale_3,velocity_combined,'Color',[0 1 1]);hold off;
legend('Velocity Reference','Velocity'); %
set(gca,'FontSize', 20); %
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Velocity(m/s)','fontsize',20);



Figure(9)
hold  on;plot(time_scale_3,x1r_combined(1:length(position_combined)),'Color',[1  0
1]);plot(time_scale_3,position_combined,'Color',[0 1 1]);hold off;
legend('Position Reference','Position'); %
set(gca,'FontSize', 20); %
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Position(m)','fontsize',20);


Figure(10)
plot(time_scale_3,x1r_combined(1:length(position_combined))-
position_combined,'Color',[1 0 1])
legend('Position Error');
set(gca,'FontSize', 20); %
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Error(m)','fontsize',20);
```

```
Figure(11)
plot(time_scale_3,x2r_combined(1:length(position_combined))-
velocity_combined,'Color',[1 0 1])
legend('Velocity Error'); %
set(gca,'FontSize', 20); %
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Error(m/s)','fontsize',20);


Figure(12)
plot(time_scale_3,position_combined,'Color',[1 0 1])
legend('Position of Train'); %
set(gca,'FontSize', 20); %
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Position(m)','fontsize',20);

Figure(13)
hold on;plot(time_scale_1,x1r,'Color',[1 0 1]);plot(time_scale_2,x1r_combined);hold
off;
legend('Initial Reference','Adjusted Reference');
set(gca,'FontSize', 20);
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Position(m)','fontsize',20);

Figure(14)
hold on;plot(time_scale_1,x2r,'Color',[1 0 1]);plot(time_scale_2,x2r_combined);hold
off;
legend('Initial Reference','Adjusted Reference'); %
set(gca,'FontSize', 20); %
xlhand = get(gca,'xlabel');
set(xlhand,'string','Time(s)','fontsize',20);
ylhand = get(gca,'ylabel');
set(ylhand,'string','Velocity(m/s)','fontsize',20);



fin_1_err=x1rhc(n_interval)-x_r(n_interval,1);
fin_2_err=x1rhc(n_interval)-x_r(n_interval,5);
fin_3_err=x1rhc(n_interval)-x_r(n_interval,9);
fin_4_err=x1rhc(n_interval)-x_r(n_interval,13);
fin_5_err=x1rhc(n_interval)-x_r(n_interval,17);
```

```
fin_6_err=x1rhc(n_interval)-x_r(n_interval,21);
variable_file=
sprintf('%s_variable_td_%d_wn_%d_md_%d_mn_%d_an_%d.mat',datestr(now,'mm
dd_HHMMSS'),delay_dev,wn_dev,dev_m,m_noise,a_noise);
save(variable_file);


%
fprintf(file_MM,'%d,%f,%f,%f,%f,%f,%f,%f,%f,%f\n'  ,ff,delay_dev,wn_dev,dev_m,
fin_1_err,fin_2_err,fin_3_err,fin_4_err,fin_5_err,fin_6_err);
fprintf(file_MM,'%f,%f,%f,%f,%f,%f,%f,%f,%f\n'  ,delay_dev,wn_dev,dev_m,fin_1_
err,fin_2_err,fin_3_err,fin_4_err,fin_5_err,fin_6_err);

% end
end
end
end

fclose(file_MM);
```

Reference

[1] Ministry of Land, Infrastructure and Transport, (2015), “Korea Railroad Statictics”, http://stat.molit.go.kr/portal/cate/statView.do

[2] KRRI, (2013), “Study on the Rapidization for the Metro Transit System and the Related Compatible Functions ”, KRRI 2013-006 (ISBN: 978-89-97667-20-8)

[3] H. A. Ahmad, (2013), "Dynamic Braking Control for Accurate Train Braking Distance Estimation under Different Operating Conditions," Virginia Polytechnic Institute and State University.

[4] S. Iwnicki, (2006),Handbook of Railway Vehicle Dynamics: CRC press.

[5] Q. Song, Y. Song, and W. Cai, (2011),"Adaptive Backstepping Control of Train Systems with Traction/Braking Dynamics and Uncertain Resistive Forces," Vehicle System Dynamics, vol. 49, pp. 1441-1454.

[6] C.-G. Kang, (2007),"Analysis of the Braking System of the Korean High-speed Train Using Real-time Simulations", Journal of Mechanical science and Technology, vol. 21, pp. 1048-1057.

[7] D. Chen and C. Gao, (2012),"Soft Computing Methods Applied to Train Station Parking in Urban Rail Transit," Applied Soft Computing, vol. 12, pp. 759-767.

[8] Q. Song, Y.-d. Song, T. Tang, and B. Ning, (2011),"Computationally Inexpensive Tracking Control of High-speed Trains with Traction/Braking Saturation," Intelligent Transportation Systems, IEEE Transactions on, vol. 12, pp. 1116-1125.

[9] D. Chen, R. Chen, Y. Li, and T. Tang, (2013),"Online Learning Algorithms for Train Automatic Stop Control Using Precise Location Data of Balises,".

[10] Z.-Y. Yu and D.-W. Chen, (2011),"Modeling and System Identification of the Braking System of Urban Rail Vehicles," Journal of the China railway Society, vol. 33, pp. 37-40.

[11] K. Teramoto, K. Ohishi, T. Kondo, S. Makishima, K. Uezono, and S. Yasukawa, (2014),"Electro-pneumatic Blended Braking Control of Regenerative Brake and Air Brake based on Estimated Adhesion Coefficient," IEEJ Journal of Industry Applications, vol. 3, pp. 75-85.

[12] H. Dong, B. Ning, B. Cai, and Z. Hou, (2010),"Automatic Train Control System Development and Simulation for High-speed Railways," IEEE Circuits and Systems Magazine, vol. 10, pp. 6-18.

[13] H. Yamazaki, Y. Karino, M. Nagai, and T. Kamada, (2005),"Wheel Slip Prevention Control by Sliding Mode Control for Railway Vehicles(Experiments Using Real Size Test)," in Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on, pp. 271-276.

[14] Joon-Hyuk Park, Byeong-Choon Goo, (2007),"Dynamic Modeling of a Railway Vehicle under Braking" The Korean Society for Railway, vol. 10, pp. 431-437.

[15] Taeyeon Lee et al., (1999),"The Selection of ATO Profile on Precision Stop Controller for Urban Railway" Journal of the Korean Society for Railway, pp. 251-258.

[16] H. Zhiwu, T. Haitao, and F. Yanfen, (2010),"The Longitudinal Dynamics of Heavy-Haul Trains in the Asynchronous Brake Control System," in Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on, pp. 900-903.

[17] YUJIN machinery Ltd., (2008), “다중 연결기(제품 규격서)”, document number:RPs-CP1-B19K

[18] Roaduser International Pty Ltd, (2000), “IN-SERVICE ASSESSMENT OF ROAD-FRIENDLY SUSPENSIONS”, Report for Australian National Road Transport Commission.

[19] 이하라 가즈오, (2012), 철도차량 메커니즘 도감, 골든벨

[20] Daniel Liberzon and A. Stephen Morse, (1999), “Basic Problems in Stability and Design of Switched Systems”, Control Systems, IEEE, vol. 10, pp.59-70.

[21] M.Wicksl, P. Peleties and R.DeCarlo, (1998), “Switched Controller Synthesis for the Quadratic Stabilisation of a Pair of Unstable Linear Systems”, European Journal of Control, vol. 4, pp.140-147.

[22] Ir. Paul B.L. Wiggenraad, (2001), “Alighting and boarding times of passengers at Dutch railway stations”, TRAIL Research School

[23] Sang-Soo Kim, Young-Guk Kim, Ki-Hwan Kim, Choon-Soo Park, (2007), “Torque Measurement of Tripod Shaft for HSR-350x”, Journal of the Korean Society for Railway
[24] Buyeon Yu, Sungmin Aum, Jungtai Kim, Yongsoon Eun, Musun Kim, Kyoungjoon Ko, (2015), “Brake Time Delay Model for Metro Train Precision Stop Control”, ICROS 2015
[25] Buyeon Yu, (2015),"Modeling and Precise Stop Control Simulator Design of Metropolitan Trains with Feedforward and PI Control”, DGIST.
[26] Joel Ames, (1913), “Train Control System”, US Patent 1053693 A
[27] Yongsoon Eun, Sungmin Aum, Buyeon Yu, Yuchang Won, (2014), “정위치 정차 제어를 위한 열차 상세 모델링 및 시뮬레이터 개발”, DGIST
[28] W.H. Kwon and S. Han, (2005), “Receding Horizon Control”, Springer
[29] W. R. Van Soest, Q. P. Chu, and J. A. Mulder, (2012), Combined Feedback Linearization and Constrained Model Predictive Control for Entry Flight, Journal of Guidance, Control, and Dynamics

요 약 문

열차 모델링과 Receding Horizon Control을 적용한 정밀 도시철도 정위치 정차 제어

근시일내 한국 도시철도에 적용될 신기종 열차의 모델링과 이에 기반하여 설계된 제어기를 시뮬레이션을 통해 검증하였다. 모델링은 한국철도기술연구원에서 제공된 실제 실험 파라미터와 기 측정된 수치를 바탕으로 진행하였다. 제어기는 두 단계로 나누어지는데, 열차 출발 시에는 feedforward 와 PI 에 기반한 제어기가 적용되고, 정차 위치 근처에 다다라 속도가 액추에이터에 끼치는 비선형성이 거의 사라졌을 때 선형 이동구간 제어기로 바뀌어 제어된다. 선형 이동구간 제어기가 도입되는 시점에 새로운 레퍼런스가 온라인으로 새로 생성된다. 적정 수준의 측정오차와 액추에이터 외란, 모델 오차등을 감안하여 시뮬레이션을 진행하였고, 이때 모두 정해진 0.1 미터 오차 안에 정차하였음을 확인할 수 있었다.

핵심어: 도시철도, 정위치 정차 제어, 열차 브레이크 모델, 이동구간 제어, model predictive control.