ROBUST CONTROLLER DESIGN OF FOUR WHEEL STEERING SYSTEMS USING MU SYNTHESIS TECHNIQUES

Author(s): X. Gao  
B. D. McVey  
R. L. Tokar

Submitted to: 34th CDC  
New Orleans, LA  
December 1995
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
Robust Controller Design of Four Wheel Steering Systems Using $\mu$ Synthesis Techniques

Xiaoming Gao*  
Brian D. McVey†  
Robert L. Tokar†

Mail Stop B261  
Los Alamos National Laboratory, University of California  
Los Alamos, NM 87545

Submitted to 34th CDC, New Orleans, Louisiana, Dec. 1995  
February 27, 1995

Abstract

In this paper, a linearized four wheel steering (4WS) system model is deduced and then modified into a form which is appropriate for applying Matlab $\mu$ Toolbox to design robust controller. Several important topics are discussed in detail, such as 1) how to make system set-up match Matlab $\mu$ Toolbox requirement, 2) how to select weights based on plant's uncertainty, 3) how to solve controller discretization problem, and 4) how to adjust the system so that the conditions necessary for using a state-space formula to solve $H_\infty$ optimal (sub-optimal) problem and performing the Matlab $\mu$ Toolbox $D-K$ iteration procedure are satisfied. Finally simulation results of robust controller and a PID controller are compared.

*Center for Nonlinear Studies/Applied Theoretical Physics Division, xgao@demos.lanl.gov  
†Applied Theoretical Physics Division.
1. Introduction

In order to increase a vehicle's maneuverability, stability, and safety, four wheel steering systems have been adopted by many Japanese and German automakers in recent years. With four wheel steering, a faster lane change or quick lateral acceleration with a less yaw rate is achieved. Overall, a properly implemented four wheel steering system can make a vehicle more maneuverable at low speeds and more responsive and stable in high speed transient maneuvers.

A survey paper by Furukawa et al. [11] summarizes the research activities of major Japanese automakers before 1989 on four wheel steering techniques, and provides a good reference for early work on the topic. In a recently published book [1] and related papers [2] [3] Ackermann discusses the basic four wheel steering systems from a feedback control point of view. In this book, the basic four wheel steering control system equations are deduced based on a bicycle model. He also develops a control law which decouples the lateral and yaw motion by using yaw rate feedback to the front wheels. He further shows that robustness with respect to uncertain operating conditions of the vehicle can be achieved. In recent papers, by Ono, Hirano et al. [22] [23] [18] [19], an integrated four wheel steering (4WS) and four wheel drive (4WD) strategy is proposed and tested in a prototype vehicle. Although the robust four wheel steering controller is designed by the $\mu$ synthesis procedure, it is developed in a disturbance rejection setting, not the more relevant tracking problem set-up. In [26] and [27], a neural network (NN) based controller is used to actively control a prototype four wheel steering vehicle. A sequence of input-output data pair collected from a real four wheel steering car is used to train the controller, which is composed of a multi-layer NN. The nonlinear relation between the slip angle of tire and the cornering force is approximated with the NN. The feedforward and feedback gain of rear wheel steering controller is determined adaptively by NN according to variation of front wheel steering angle. In [17], optimal control theory is used to minimize the sideslip angle at the center of gravity of the vehicle. For other references about four wheel steering systems, the interested reader can also see [5], [20], [29], and [30].

In this paper, a linearized four wheel steering model is deduced based on a bicycle model of the vehicle. The model is placed into a form which fits the $\mu$ analysis and synthesis framework of the Matlab $\mu$ Toolbox. By carefully selecting the weights, the controller obtained is robust with respect to all operating conditions. Problems encountered during the design are explained in detail, which include 1) how to make system set-up match Matlab $\mu$ Toolbox requirement, 2) how to select
weights based on system model, 3) how to solve the stability problem caused by discretization of reduced order continuous controller, and 4) how to adjust the system so that the conditions necessary for using a state-space formula to solve $H_{\infty}$ optimal (sub-optimal) problem and performing the Matlab $\mu$ Toolbox $D-K$ iteration procedure are satisfied. Finally, simulation results of a PID controller and the robust controller are compared.

2. A Simplified Four Wheel Steering Car Model

The essential features of car steering dynamics in a horizontal plane can be described by a bicycle model as shown in figure 2.1 [1]. In the figure, $\delta_f$ and $\delta_r$ are the front and rear steering angles, $f_f$ and $f_r$ are forces transmitted from the road surface via the front and rear patches to the car body, $\beta$ is the vehicle sideslip angle, CG is the center of the gravity of the car which has velocity $v$, $l_{wb} = l_f + l_r$ is the wheel base, $(x_0, y_0)$ is an inertially fixed coordinate system, and $(x, y)$ is chassis coordinate system fixed with the vehicle that rotates with a yaw rate $R = \dot{\psi}$ with respect to $(x_0, y_0)$.

The tire side forces can be projected through the steering angle into chassis coordinates $(x, y)$, thus the longitudinal force and lateral force caused by front and rear tires can be expressed as

\[ F_z = -f_f \sin \delta_f - f_r \sin \delta_r \]  
\[ F_y = f_f \cos \delta_f + f_r \cos \delta_r \triangleq F_f + F_r \]  

From figure 2.1, the following three equations are obtained which characterize the motion of the vehicle in the horizontal plane [16] [1]:

\[ \]
longitudinal motion

$$-mv(\dot{\beta} + R) \sin \beta + mv \cos \beta = F_x$$  \hspace{1cm} (2.3)

lateral motion

$$mv(\dot{\beta} + R) \cos \beta + mv \sin \beta = F_y$$ \hspace{1cm} (2.4)

yaw motion

$$I \ddot{R} = l_f F_f - l_r F_r$$ \hspace{1cm} (2.5)

where \(v = |v|\), \(I\) is yaw moment of inertia, and \(m\) is vehicle mass.

If \(\beta\) is small, and \(v\) is varying slowly, the plane motion equations can be rewritten as

$$mv(\dot{\beta} + R) = F_y = F_f + F_r$$ \hspace{1cm} (2.6)

$$I \ddot{R} = l_f F_f - l_r F_r.$$ \hspace{1cm} (2.7)

These equations are the starting point for four wheel steering problem used by many researchers [22] [23] [18] [19] [1] [2] [3].

Following [1] [12] [7], the front and rear tire sideslip angles are defined as

$$\alpha_f = \delta_f - \beta_f$$

$$\alpha_r = \delta_r - \beta_r$$

where \(\beta_f\) and \(\beta_r\) represent front and rear chassis sideslip angles respectively.

Generally, the tire model is represented by nonlinear relation between tire side force and tire sideslip angle. Within the linear region, the nonlinear tire characteristics can be approximated as

$$f_f = \mu C_f \alpha_f$$ \hspace{1cm} (2.8)

$$f_r = \mu C_r \alpha_r$$ \hspace{1cm} (2.9)

$$C_f \triangleq k_{cf} C_{fn}(mg \frac{l_r}{l_{wb}})$$

$$C_r \triangleq k_{cr} C_{rn}(mg \frac{l_f}{l_{wb}})$$

where \(\mu\) is the adhesion coefficient between road surface and the tire ranging from 1 (dry road) to 0.15 (ice road), and \(C_f\) and \(C_r\) are cornering stiffness of front and rear tire respectively, \(C_{fn}\) and \(C_{rn}\) are normalized cornering stiffnesses, \(k_{cf}\) and \(k_{cr}\) are cornering stiffness coefficients.
When $\beta, \beta_f$, and $\beta_r$ are assumed very small, (corresponding active rear steering angle $\delta_r \leq .017$ radian $\approx 1$ degree [23]), that is, within the linear region of tire characteristics, the dynamic equations 2.6 and 2.7 can be expressed in a linear form [1]. Following Appendix A of [1], and using equations 2.8 and 2.9, yields the linearized 4WS model. That is, the transfer function from front and rear steering input to yaw rate can be expressed as [18] [1]

$$R_L(s) = \frac{b_1 s + b_2}{s^2 + a_1 s + a_2} \Delta_f(s) + \frac{b_3 + b_4}{s^2 + a_1 s + a_2} \Delta_r(s)$$

$$\Delta = T_f(s) \Delta_f(s) + T_r(s) \Delta_r(s)$$

(2.10)

where $\Delta_f(s)$ and $\Delta_r(s)$ are Laplace transform of $\delta_f$ and $\delta_r$ respectively,

$$a_1 = \frac{C_f + C_r}{mv} \mu + \frac{l_f C_f + l_r C_r}{I v} \mu,$$

$$a_2 = -\frac{1}{I} C_f \mu + \frac{(l_f + l_r)^2 C_f C_r}{I m v^2} \mu^2,$$

$$b_1 = -\frac{l_f C_f}{I} \mu,$$

$$b_2 = \frac{(l_f + l_r) C_f C_r}{I m v} \mu^2,$$

$$b_3 = -\frac{l_r C_r}{I} \mu,$$

$$b_4 = -b_2.$$  

(2.11)-(2.15)

and

$$R_0(s) = \frac{k b_2}{(1 + T \theta) a_2} \Delta_f(s) \Delta R(s)$$

(2.16)

(2.17)

The desired yaw rate is assumed to be [18]

which is corresponding to zero chassis sideslip [11].

Finally, the four wheel steering problem can be simplified as a yaw rate tracking problem as shown in figure 2.2. where $K$ is the controller to be designed, $T_\theta$ is the transfer function of rear wheel steering actuator.

From expressions of $a_i$ and $b_i$ in linearized 4WS model, it is obvious that $T_f(s)$, $T_r(s)$, and $T_R(s)$ are the function of velocity $v$, mass $m$, cornering stiffness $C_f$ and $C_r$, which may vary at different driving conditions. Therefore, a controller is needed in order to satisfy all robust requirements with respect to all uncertainties.
3. Controller Design By Using Matlab $\mu$ Toolbox

In this section, we first modify the linearized 4WS system model into a form suitable for the $H_\infty$ control/$\mu$ analysis and synthesis framework as required by Matlab $\mu$ Toolbox software package [4]. Then we discuss selection of weighting functions based on the structure and parameter distribution of the system. Finally, we will discuss a few important technical problems encountered in the design. These problems include controller order reduction and discretization, and the adjustment of system set-up. The system set-up needs adjustment due to following two reasons: one is for the purpose of applying 'dkit' command, the other is for improving the performance of discretized reduced order controller. In other words, in order to apply 'dkit' command, we need to adjust system set-up such that all those necessary assumptions [13] [9] [21] [14], which makes $H_\infty$ control algorithm executable to our system are satisfied. Because controller order reduction and discretization might deteriorate the performance capability of a final reduced order discrete controller, the adjustment of the system set-up also becomes necessary during the design.

3.1. Model modification for $H_\infty$ control and $\mu$ design

To fit the 4WS system model into the framework of $H_\infty$ and $\mu$ theory, the model illustrated in figure 2.2 is modified into a equivalent form as shown in figure 3.1 where

$$T_{f0} \triangleq T_f(v_0)$$

$$T_{r0} \triangleq T_r(v_0)$$
Figure 3.1: A simplified four wheel steering system model fit in a $H_\infty$ and $\mu$ frame work.

are the nominal front and rear wheel transfer function blocks respectively, which are obtained based on equation 2.10 by taking all parameters at their nominal values ($v = v_0 = 35 \text{ m/s}$). For simplicity, we treat all parameters of $a_i$ and $b_i$ as constants except the velocity $v$ since it dominates the uncertainty. The blocks $\Delta_f$ and $\Delta_r$ are the corresponding plant structured uncertainty perturbations [10] to nominal transfer function $T_{f0}$ and $T_{r0}T_a$ respectively. The weighting function $W_{11}$ and $W_{12}$ represent the uncertainty frequency response of the corresponding transfer function blocks. Here, $v_0$ is used to denote nominal speed. Additional measurement noise, external disturbance, and uncertainty blocks can be added to the model depending on the situation and practical needs if a more complicated model is necessary to the design. To simplify the case, we assume yaw rate reference model is fixed as

$$T_{R0}(s) \triangleq T_R(s).$$

Figure 3.1 can be rewritten in a familiar form as shown in figure 3.2, where $\Delta_{fuc}$ is a artificially constructed fictitious uncertainty block, which is used for the purpose of applying the Main Loop Theorem which provides a robust performance test [24] [25] [28]. The system set up can be expressed in a more compact form as shown in Figure 3.3, which is frequently seen in $H_\infty$ control and $\mu$ analysis literature. More explicitly, for the ease of using Matlab $\mu$ Toolbox, figure 3.2 can be alternatively expressed as shown in figure 3.4, where the prefix sys on each
Figure 3.2: A simplified 4WS model for the purpose of applying Matlab μ Toolbox in $H_\infty$ and μ design.

block implies that each block now is expressed in system matrix form as required by Matlab μ Toolbox. Based on the configuration of figure 3.4, the system can be easily implemented by Matlab command source code following the instructions from Matlab μ Toolbox Manual.

3.2. Weighting function selection

Generally speaking, a frequency dependent weighting function matrix

$$W_1 \triangleq \begin{bmatrix} W_{11} & 0 \\ 0 & W_{12} \end{bmatrix}$$

is used for covering or representing all possible structured uncertainty distributions between each realizable system transfer function block and its corresponding nominal value transfer function block. $W_2$ is used for specifying systems performance. Note that in our simplified system model, actuator transfer function $T_a$ and rear wheel input reference model $T_{R0}$ are assumed to be fixed.
Based on our multiplicative uncertainty assumptions to front wheel and rear wheel systems, we have

\[
T_f = (1 + \Delta_f W_{12}) T_{f0} \\
T_f T_a = (1 + \Delta_r W_{11}) T_{a0} T_a
\]

where \(\|\Delta_f\|_{\infty} \leq 1\) and \(\|\Delta_r\|_{\infty} \leq 1\). Taking the front wheel system as an example, weighting function \(W_{12}\) can be deduced as shown in the following. Rewrite the above first equation as

\[
\frac{T_f - T_{f0}}{T_{f0}} = \frac{T_f(v) - T_f(v_0)}{T_f(v_0)} = \Delta_f W_{12},
\]

taking the \(H_\infty\) norm and considering \(\|\Delta_f\|_{\infty} \leq 1\), yields

\[
\left| \frac{T_f(v) - T_f(v_0)}{T_f(v_0)} \right| \leq |W_{12}(j\omega)|.
\]

In other words, \(W_{12}\) can thus be determined based on following algorithm:
Figure 3.4: System set up required by Matlab μ Toolbox software package.
$$|W_{12}(j\omega)| \geq \left\{ \max_{v=S} \left| \frac{T_f(v) - T_f(v_0)}{T_f(v_0)} \right| ; S \triangleq [0, 50] \text{ (m/s)} \right\}. \tag{3.1}$$

Note that the weighting function $W_{12}$ is used to normalize the size of the unknown perturbation $T_f - T_{f0}$ with respect to $T_{f0}$. At any frequency $\omega$, the magnitude $|W_{12}(j\omega)|$ can be interpreted as the magnitude distribution of uncertainties of the model $T_f(v)$ with respect to selected nominal model $T_f(v_0)$, expressed as a percentage. Similarly, we have

$$|W_{11}(j\omega)| \geq \left\{ \max_{v=S} \left| \frac{T_r(v) - T_r(v_0)}{T_r(v_0)} \right| ; S \triangleq [0, 50] \text{ (m/s)} \right\}. \tag{3.2}$$

$W_2(j\omega)$ is the weight specifying system performance. It can be deduced such that the system’s tracking error $e$ is bounded as $[8, 15]$

$$\|e\|_{\infty} = \left\| \frac{1}{1 + PC} r \right\|_{\infty} < \frac{1}{\|W_2\|_{\infty}}. \tag{3.3}$$

The function of $W_2$ is quite clear, the system tracking error $e$ has an upper bound of $1/\|W_2\|_{\infty}$. Note that $\|W_2\|_{\infty}$ is frequency dependent. Based on system performance specification, suitable weighting function $W_2(j\omega)$ can be selected during design procedure, such that the controller can obtain the required robust performance.

**Remark:** Based on inequality 3.3, the robust performance capability of Hirano’s 4WS robust controller can be easily verified. For the details of this problem, cf. [22], [23], [18], and [19]. The $W_2$ selected by Hirano is [19]

$$W_2 = \frac{1}{5s + 1}. \tag{3.4}$$

(Note, in his paper Hirano uses $W_s$ to denote $W_2$). If we substitute this $W_2$ into inequality 3.3, the system steady state tracking error (when letting $s = 0$) will be 1, which corresponds to a possible 100% tracking error! In their another recent publication [22], they select

$$W_2 = \frac{3}{5s + 1}. \tag{3.5}$$

the corresponding system steady state tracking error is still very high, up to $1/3 \approx 33\%$. The conclusion is that the simulation result shown in [22] [23] [18] [19] is not due to $\mu$ theory designed 4WS robust controller, but due to 4WD system control.\]
which dominates the dynamic behavior of Ono and Hirano's integrated 4WS-4WD system. The set-up Hirano and Ono use in their design is for disturbance rejection problem, whereas the 4WS problem is a tracking problem.

The author's experience shows that $W_{11}$ and $W_{12}$ obtained by directly using the inequalities 3.2 and 3.1 can not be accepted because the structured singular value by $D - K$ iteration is much greater than 1. It has been pointed out in [2] that the 4WS system is not controllable for $v = 0$ since, in that situation, the system's uncertainty becomes infinity. As in [2], we assume $v > 0$. Reducing uncertainty in $v$ from $0^+ \sim 50$ m/s to $5 \sim 50$ m/s, the corresponding magnitude Bode plot of

$$\max \left| \frac{T_f(v) - T_f(0)}{T_f(0)} \right|_v \quad \text{and} \quad \max \left| \frac{T_r(v) - T_r(0)}{T_r(0)} \right|_v$$

(3.4)

are still too large to be accepted. The plots of Eqn. 3.4 when $v = 5 - 50$ meter/sec are as shown in figure 3.6 and 3.5.

From the plots, it is clear that the peak value of the normalized uncertainty at this situation is about $-2dB \simeq 0.8$ which occurs at about $10$ rad/sec $\simeq 1.5$ Hz. This means that when $v$ is varying between $5$ m/s to $50$ m/s, the uncertainty is as high as $80\%$ within the system's working frequency range with respect to nominal velocity $35$ m/s.

In order to find an envelope of the uncertainty, a second order transfer function

$$wtf = \frac{k(s + a)(0.01s + 1)}{(s + b)(s + c)}$$

is used as a design tool to approximate the envelope of the normalized uncertainty. This is implemented by Matlab m-file function $\text{wtrta.m}$ and $\text{wtf.m}$ in a form

$$[n_{11}, d_{11}] = \text{wtrta}(k, a, b, c)$$

and

$$[n_{12}, d_{12}] = \text{wtf}(k, a, b, c),$$

where $W_{11} \triangleq n_{11}/d_{11}$, $W_{12} \triangleq n_{12}/d_{12}$.

Based on authors design, the normalized uncertainty plot of

$$\left\{ \max \left| \frac{T_f(v) - T_f(v_0)}{T_f(v_0)} \right|_{v = S} ; S \triangleq [v_{\text{init}}, v_{\text{final}}] \text{ (m/s)} \right\}$$

(3.5)

and

$$\left\{ \max \left| \frac{T_r(v) - T_r(v_0)}{T_r(v_0)} \right|_{v = S} ; S \triangleq [v_{\text{init}}, v_{\text{final}}] \text{ (m/s)} \right\}$$

(3.6)
Figure 3.5: The Bode plot (magnitude) of normalized plant uncertainty used for finding $W_{11}$.

Figure 3.6: The Bode plot (magnitude) of normalized plant uncertainty used for finding $W_{12}$. 
are implemented by Matlab m-file \texttt{tfenvelop.m} and \texttt{trapenvelop.m}, where \([v_{init}, v_{final}] \subseteq [5, 50] \text{ m/s} \) are selected such that the magnitude Bode plot of \(W_{11}\) and \(W_{12}\) produced by the design tools \texttt{wtrta} and \texttt{wtf} cover the magnitude Bode plot of Eqns 3.5 and 3.6 respectively, and \(v_0\) is the selected nominal velocity, which is selected in such a way that it makes the peak value of Eqns 3.5 and 3.6 as small as possible. The final selection of weights needs to be verified by the Matlab \(\mu\) Toolbox design. If the result is not satisfactory, the weights need to be changed.

Finally, we get

\[
W_{11} = \frac{0.0065s^2 + 6.5111s + 11.05}{s^2 + 15.1s + 53.2}
\]

and

\[
W_{12} = \frac{0.005s^2 + 5.0095s + 9.5}{s^2 + 14.8s + 48}.
\]

The magnitude Bode plot of \(W_{11}\) and \(W_{12}\) are as shown in figure 3.7 and figure 3.8 respectively.

Remark: The controller obtained through \(D - K\) iteration seems to only guarantee the stability within the range \(v \in [v_{init}, v_{final}]\), actually the controller works fine even at very low speeds such as \(v = 1 \text{ m/s} \ll v_{init}\).

The selection of \(W_2\) in our problem is rather simple, since there are several basic constraints we can follow.

The first constraint is that we want a low order weight \(W_2\), since the order of the controller obtained by using Matlab \(\mu\) Toolbox is directly related to that of the weights selected. The lowest possible order of the controller obtained is equal to the sum of the original system's order and the order of the weights used during the design. So we want the weight selected have the lowest order possible.

The second constraint is based on the steady state tracking error of the system. Fortunately system's steady state tracking error is upper bounded by \(1/\|W_2\|_\infty\) when \(s = j\omega = 0\).

The third constraint is the bandwidth of the weight \(W_2\) selected. The bandwidth of \(W_2\) must comparable to that of system's working range. Obviously, we do not need to select a weight \(W_2\) which has a wider bandwidth than that of the system's working range. On other hand, we hope that the bandwidth of selected weight \(W_2\) will be not too small compared with system's bandwidth, otherwise the performance of the controller will deteriorate at frequencies of the high end of the system's working range.

A final constraint is that the relative degree of \(W_2\) be zero, that is, we want the numerator and denominator of \(W_2\) be same order. The reason is that when
Figure 3.7: The magnitude Bode plot of weighting function $W_{11}$.

Figure 3.8: The magnitude Bode plot of weighting function $W_{12}$. 
the relative degree of $W_2$ is larger than zero, the necessary conditions for applying $H_{\infty}$ optimal or suboptimal algorithm (or say, state-space or two Riccati equation methods) as shown in [9] and [13] are not necessarily satisfied. Usually the order of $W_2$ should be no more than two with zero relative degree.

Based on above general criterion, $W_2$ is selected as

$$W_2 = \frac{0.4(s + 3)}{(s + 0.03)}.$$  

The corresponding steady state tracking error is less than $1.2/0.03 = 2.5\%$. The magnitude Bode plot of $W_2$ is as shown in figure 3.9.

Using selected weights, we get a $25^{th}$ order controller by running dkit command. The controller can be reduced to $6^{th}$ order by balanced truncation model reduction approach [14].

### 3.3. Controller discretization and related problems

In practice, the controller is used in discretized form because it is implemented by digital computer. Usually by using bilinear transformation, or say, Tustin's methods, the discretized approximation $K_T(z)$ of a continuous transfer function $K(s)$ can be obtained:

$$K_T(z) \approx K(s)|_{s=(2/\tau)((z-1)/(z+1))},$$

where $T$ is the sampling time interval. The relation of $K(s)$ and $K_T(z)$ can be expressed as shown in figure 3.10. Thus our system becomes a sampled-data system as shown in figure 3.11.

The continuous controller performance is badly deteriorated when discretized directly by using the bilinear transformation

$$s = \frac{2(z - 1)}{T(z + 1)}. \quad (3.7)$$

Actually, when the directly discretized controller $K_T(z)$ is being used, which is obtained via directly applying bilinear transformation 3.7 to continuous controller $K(s)$, the system as shown in figure 3.11 becomes unstable at some cases. This instability is due to the introduction of the sampler and Zero-Order Hold (ZOH) into the system when the simulation are performed by Matlab Simulink Toolbox.

Based on classic control theory, there are two approaches can be used to solve the controller discretization problem. The first one is to discretize all system
Figure 3.9: The magnitude Bode plot of weighting function $W_2$.

Figure 3.10: Approximation of a continuous controller by bilinear transformation.
components and design the controller in the z domain. Considering that discrete version Matlab μ Toolbox is not available, (this is not surprising, the development of discrete control theory is far behind continuous control, because of many theoretical problems.) a second approach of analogue design of discrete controller is investigated.

Considering the existence of the sampler and ZOH during the discretization process, the system’s set-up is modified as follows. The series connection of the leading sampler and the ZOH can be approximated, by using first order Padé approximation, by a continuous transfer function

\[ G_{\text{appr}} = T^{-1} \frac{2T}{T_s + 2} = \frac{2}{T_s + 2} \]  

(3.8)

where \( T \) is the sampling interval, \( T^{-1} \) represents the transfer function of the sampler, and \( \frac{2T}{T_s + 2} \) is Padé’s first order approximation of the zero-order hold (cf. chapter 5 of [31]).

The next step is to make the system’s set-up approximate the practical configuration of the system as closely as possible when a discretized controller is being used. In other words, we hope that the system’s set-up can reflect the existence of sampler and ZOH without increasing the system’s complexity. Based on this general idea, the modified system set-up is as shown in figure 3.12.

Now the continuous transfer function \( G_{\text{appr}} \), which is the approximation of sampler and ZOH, is merged together with the plant. That is, \( G_{\text{appr}} \) is now considered as part of the plant during the design. Due to the discretization, the plant to be controlled now has an additional delay. More specifically, the transfer function block corresponding to weight \( W_{11} \) used to be \( T_\alpha T_a \), now becomes \( T_\alpha T_a G_{\text{appr}} \).

Since block \( T_\alpha T_a \) is changed, it seems that corresponding weight \( W_{11} \) should be adjusted. If we select the sampling rate as 100 Hz which is fast enough based on Shannon Sampling Theorem, considering that our controller working frequency is less than 10 Hz and the constraint of hardware implementation, then sampling interval becomes \( T = 10 \text{ ms} = .01 \text{ second} \). Thus based on the approximation of Eqn. 3.8, we have

\[ G_{\text{appr}} = \frac{2}{.01s + 2} \]  

(3.9)

The magnitude Bode plot of \( G_{\text{appr}} \) now is almost equal to 1 when \( s = j\omega \) is less than 10 Hz \( \approx 63 \text{ rad/sec} \). Thus \( W_{11} \) is unchanged in the modified system set-up. Note that \( G_{\text{appr}} \) used in our final system set-up is as shown in Eqn. 3.9, which corresponds to sampling time of 10 ms.
Figure 3.11: Simulation of sampled-data system.

Figure 3.12: Modified system set-up due to direct discretization of continuous controller obtained through Matlab μ Toolbox design.
The corresponding continuous controller after balanced order reduction and discretization works satisfactorily when the discretizing frequency $f_{ds} \geq 125$ Hz. In other words, when controller is discretized at a rate less than or equal to 8 ms, the discrete controller works very well. Based on our understanding, the inconsistency between sampling rate for system set-up during the design and that for controller discretization is due to the inaccuracy of first order Padé approximation.

3.4. System set-up adjustment

In this sub-section, we discuss a situation, which might be frequently happen when commands hinfsyn or dkit of the Matlab $\mu$ Toolbox are being executed. When we start running hinfsyn or dkit, following problem might occur. Matlab generates the following error/warning messages, for example, "matrix d21 is not in full row rank, selected $\gamma$ upper bound is too small". In this case, no matter how large you increase $\gamma$, the error message is not eliminated. The error messages in your case might be different, depending upon the situation. In most cases, if the system set-up is correct, then, this message tells you two things.

First there is no solution to your problem by the optimal/sub-optimal $H_\infty$ control algorithm, which is implemented/executed by command hinfsyn or dkit. You must modify your system set-up.

Second, at least one of the necessary conditions used for executing hinfsyn or dkit is violated due to your system set-up \[13\] [21].

Based on author's design practice, if the system's set-up does not satisfy those assumptions required by the state-space or two Riccati equation methods in $H_\infty$ optimal or suboptimal control design, then the only possible way to satisfy the assumptions is to adjust the relative degree of the transfer function block within the system set-up configuration. The structure of the system set-up and the numerical parameters of the components in the system are very hard to change due to the performance constraint on the system.

This type of adjustment has been used by the control engineer for a long time. The most well known example is the implementation of PID or PD controller. It is well known that a true Derivative controller $k_D s$ can not be implemented as a causal system, instead, an approximation $k_D = s/(1 + \tau s)$, where $\tau$ is a very small number such as $\tau = .001$, are being used.

Similar techniques are adopted in our design. When 'dkit' command is being executed where

$$W_1 = \text{diag}\{W_{11}, W_{12}\}$$
and $W_2$ are selected as discussed in a previous subsection, and system set-up is based on Fig. 3.12, the following error message appears: "matrix d21 is not in full row rank". At this situation, control design can not be continued. The way the author eliminated this error message is to adjust system set-up in such a way that adjustment effectively eliminates those factors causing error messages, and at same time does not modify the dynamic properties of the original system. More specifically, during the design, we adjust rear wheel reference model transfer function block $T_{R0}$ from

$$T_{R0} = \frac{230.3138}{1.9644s + 65.4791}$$

to

$$T_{R0} = \frac{0.001s + 230.3138}{1.9644s + 65.4791},$$

which has not changed the original system's dynamic property within our controller's working frequency range 0-5 Hz. The controller thus obtained works excellent and is not affected by such adjustment. The corresponding closed system $\mu$ plotting is as shown in Fig. 3.13.

4. Simulation Results

Two different types of input signals $\delta_f$ were tested: a step input, and sinusoidal signals of 0.5, 1, 2, 5, 10, 20, 50, 100 Hz, all with same magnitude, $\delta_f = 1^\circ$. The system is tested at different speeds ranging from $v = 1 \text{ m/s}$ to $v = 50 \text{ m/s}$ and with two different masses, all other parameters are fixed. In one situation, we assume $m = m_0 = 1714.8 \text{ kg}$. In another situation, we assume system mass $m$ increasing 25%, that is, $m = 2143.5 \text{ kg}$.

The simulations shows that the controller thus obtained behaves satisfactorily in all cases.

As a comparison, the control results of a $PID$ controller are also illustrated for several cases. A $PID$ controller can implemented as

$$PID = K_p + \frac{K_i}{s} + \frac{K_d}{\tau s + 1} s$$

where we select $K_p = -0.2$, $K_i = -6$, $K_d = -0.02$, and $\tau = 0.001$, which are the best parameters obtained for our case. The discretized $PID$ controllers are obtained by using bilinear transformation (Eqn. 3.7) with sampling-time equal to 8 ms, which is same as that for robust controller.
Remark: If alternative weights and $G_{app}$ used in design are more carefully selected, or a larger value of $T$ in Eqn. 3.8 is used, we are expecting that continuous controller thus obtained after order reduction can be discretized at a larger time interval.

4.1. Summary of the simulation results

For the sake of saving space, only simulation results corresponding to $m = 1.25*m_0$ and controller discretizing sampling-time= 8 ms are listed, since these situations correspond to a worse case. Only results corresponding to 0.5, 1, and 2 Hz are listed, since other higher frequency input are unlikely. The results corresponding to higher frequencies are all stable and with acceptable accuracies. For the detail of simulation plots, cf. [15].

4.1.1. Step input

The simulation results are summarized as shown in following tables when $\delta_f$ is a step input signal:

<table>
<thead>
<tr>
<th>$\delta_f$</th>
<th>Speed (mph)</th>
<th>Settling Time</th>
<th>Overshoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>0.25 sec</td>
<td>13.6 %</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>$\leq 0.25$ sec</td>
<td>11 %</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.5 sec</td>
<td>13.3 %</td>
<td></td>
</tr>
</tbody>
</table>

As a comparison, the simulation results of PID controller are listed in following corresponding tables:

<table>
<thead>
<tr>
<th>PID controller</th>
<th>Speed (mph)</th>
<th>Settling Time</th>
<th>Overshoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>1.3 sec</td>
<td>35.8 %</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>$\geq 0.75$ sec</td>
<td>28.5 %</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1 sec</td>
<td>19.5 %</td>
<td></td>
</tr>
</tbody>
</table>

4.1.2. Sinusoidal input

The tracking error corresponding to sinusoidal input are summarized in following tables when input $\delta_f$ is a sinusoidal signal 1°.
Robust controller obtained by using Matlab $\mu$ Toolbox design and discretized at 8 ms

<table>
<thead>
<tr>
<th>Frequency</th>
<th>speed (mph)</th>
<th>0.5 Hz</th>
<th>1 Hz</th>
<th>2 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 Hz</td>
<td>112</td>
<td>2.3 %</td>
<td>4.6 %</td>
<td>8.8 %</td>
</tr>
<tr>
<td>1 Hz</td>
<td>67</td>
<td>4 %</td>
<td>3.7 %</td>
<td>15 %</td>
</tr>
<tr>
<td>2 Hz</td>
<td>20</td>
<td>1.25 %</td>
<td>2.4 %</td>
<td>4.8 %</td>
</tr>
</tbody>
</table>

For comparison, the corresponding control results of PID controller are listed in following tables:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>speed (mph)</th>
<th>0.5 Hz</th>
<th>1 Hz</th>
<th>2 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID controller</td>
<td>112</td>
<td>6.9 %</td>
<td>13.5 %</td>
<td>36 %</td>
</tr>
<tr>
<td>discretized at 8 ms</td>
<td>67</td>
<td>11.3 %</td>
<td>28.5 %</td>
<td>40 %</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2.8 %</td>
<td>6 %</td>
<td>18 %</td>
</tr>
</tbody>
</table>

5. Future Work

The discretization problem is worth further investigation. For the details, the interested reader can refer to [6]. More complicated dynamic car model should be used in our final simulation instead of present model. Considering the facts that in most cases the velocity $v$ of the car to be designed are measurable, and the information of road surface conditions are unknown, the work in next stage might be described as follows. Assuming velocity $v$ is known, design a robust controller which can accommodate big variation on road surface conditions such as tire parameter cornering coefficients $C_{fn}$ and $C_{rn}$. Since the linearized model used before now does not apply to such situation, the completely new mathematical car model deduced from car dynamics should be investigated.

Acknowledgment

The valuable comments and suggestions during the work from Prof. Roy Smith of University of California at Santa Barbara through email are greatly appreciated.

References


[23] Eiichi Ono et al., "Robust Coordinated Control for 4-Wheel-Steering and 4-Wheel-Drive Vehicle", IFAC 12th World Congress, Sydney, Australia, 1993.


Figure 3.13: The $\mu$ plot of corresponding closed loop system.