Applied Control Coursework 1

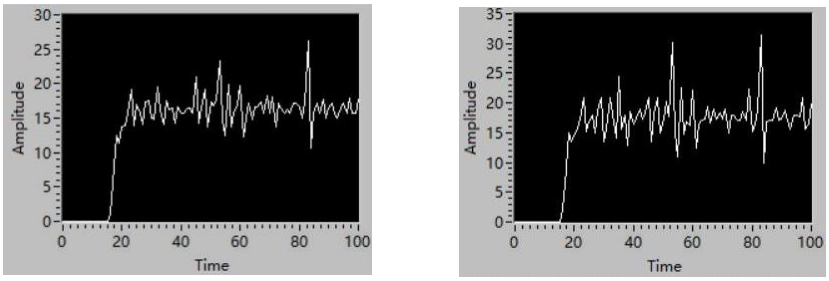
9971603 Yi Zhou

Team: 26 , Team Members: Weilin Zhong, Yi Zhou

Task 3 Sampling time and steady state characteristics

Step 1 Determine the sampling time

Too large sampling time may lead to the loss of information. It should also be noticed that the sampling time cannot be too small due to the computation time of controller and the response of actuator. In general case, to capture the principle dynamics response, the sampling time should be around 1/12 to 1/6 of the settling time.



(a) Right Motor **(b)Left Motor**

Figure 1 The response when the sampling time is 20ms and the input is 1

Set a default sampling time as 20 ms and input signal as 100% of the voltage, then the responses of the angular speeds are recorded. From the figure above, the settling time for this system is around 25x20=500 milliseconds. Thus the suitable sampling time can be selected as 80 ms.

Step 2 The steady state characteristics

If the system is linear, it should obey the rule of superposition, i.e. y(ku) should equals to k*y(u). So test the system with the input u from 0.1 to 1 with interval of 0.1, and record the corresponding steady state output y. The results are shown below. From the figure, it can be inferred that the relationship between the input percentages of voltage and output angular velocity can be modeled as a nonlinear, i.e. piece-wise, function. At low voltage of the input, i.e. 0 to 0.3, the response curve is flat. The back emf and static friction can contribute to this. Then as the input increases, the output tends to increases linearly.

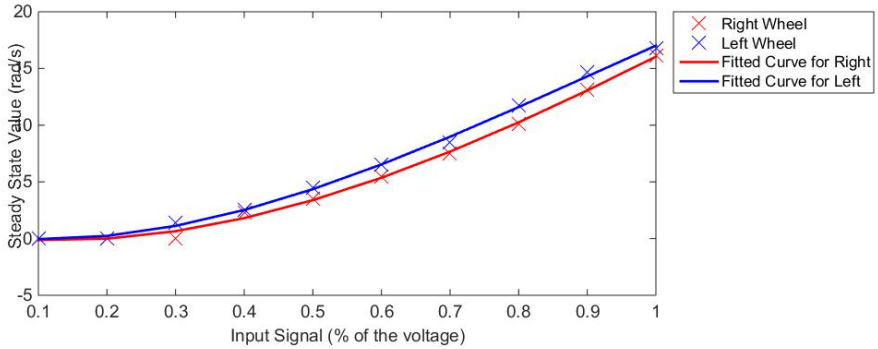


Figure 2 The nonlinear relationship between the input and output

Task 4 System Identification and Model Validation for DaNI robot

Step 1 System identification

Design the input signal as PRBS to stimulate more frequencies of the model. Using the above sampling time, run the system and record the corresponding output. We have selected 1500 data to avoid the high bias.

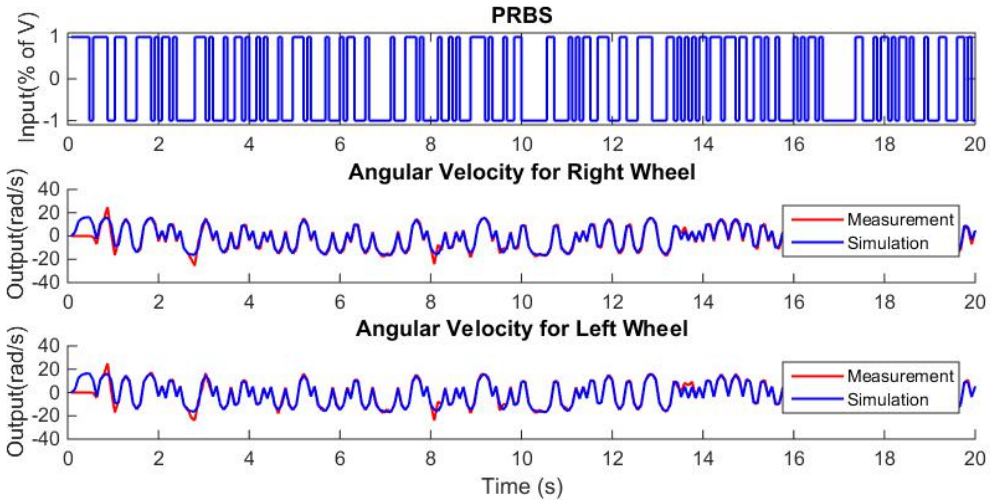


Figure 3 the input and output signal for system identification

Then, select some candidate order of na and nb. Using the least square method to minimize the mean square error of the fitting model. The corresponding model parameter and the mean square error are recorded in the table below.

na	nb	Right			Left		
		A	B	MSE	A	B	MSE
2	1	[1 -0.62 0.19]	[0 3.06]	70.98	[1 -0.54 0.12]	[0 2.75]	88.28
2	2	[1 -0.31 0.04]	[0 3.15 8.49]	8.74	[1 -0.28 0]	[0 2.80 9.17]	9.46
3	1	[1 -0.62 0.22 -0.04]	[0 3.06]	70.85	[1 -0.54 0.13 -0.03]	[0 2.75]	88.21
3	2	[1 -3.04 0.037 0.01]	[0 3.16 8.51]	8.73	[1 -0.28 0 0]	[0 2.81 9.18]	9.47
3	3	[1 -0.28 0.03 0.01]	[0 3.16 8.58 0.22]	8.73	[1 0.07 -0.08 0]	[0 2.76 10.15 3.59]	9.28

Table 1 The mean square error of candidate models

As the table shows, the order of (2,2) is best fit to this system. The higher order cases have little improvement and tends to over-fit the data. The lower order have large MSE.

Finally, we can derive the model of left and right actuators as

$$y_L(k) - 0.2767 \cdot y_L(k-1) + 0.0033 \cdot y_L(k-2) = 2.7981 \cdot u_L(k-1) + 9.1688 \cdot u_L(k-2) + \varepsilon$$

$$y_R(k) - 0.3059 \cdot y_R(k-1) + 0.0411 \cdot y_R(k-2) = 3.1468 \cdot u_R(k-1) + 8.4937 \cdot u_R(k-2) + \varepsilon$$

Step 2 Model Validation

Test the fitted model and the real system with some steps. The experiment results are shown below. From the figure, we can find that the model fit worse when the input decreases. The nonlinearity of the model may cause these discrepancies.

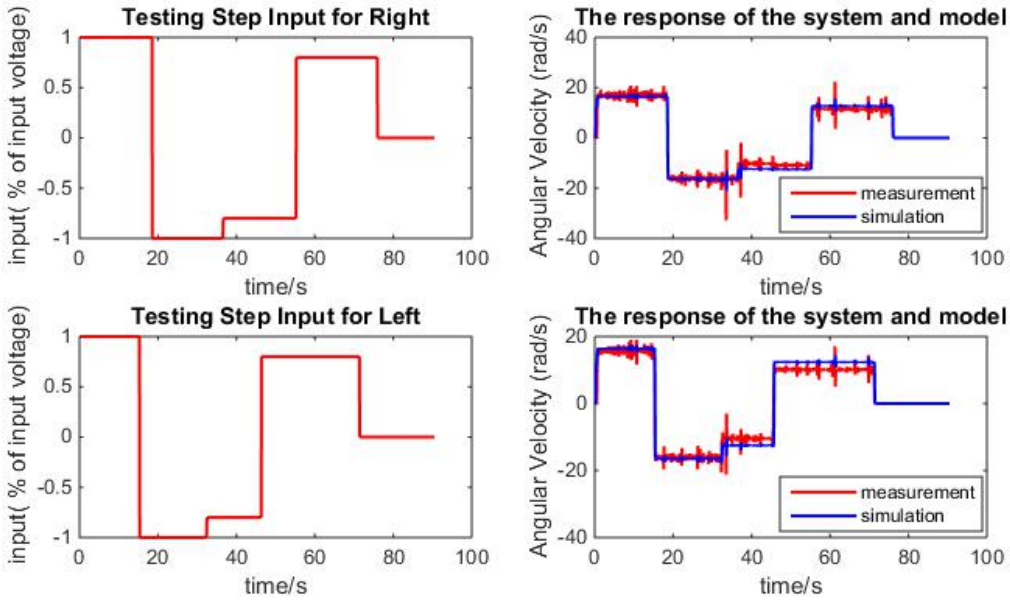


Figure 4 The simulation and real system's responses of several step inputs

The residuals are computed separately for this four steps. It can be observed that there are some outliers which may be caused by the noisy measurement. For a appropriate modeling of noise, the residual should be white noise. Thus the autocorrelation results should have only one peak at t=0. For this case, it seems there are several small peaks. Other parametric models like ARX, OE should be tested for further studies.

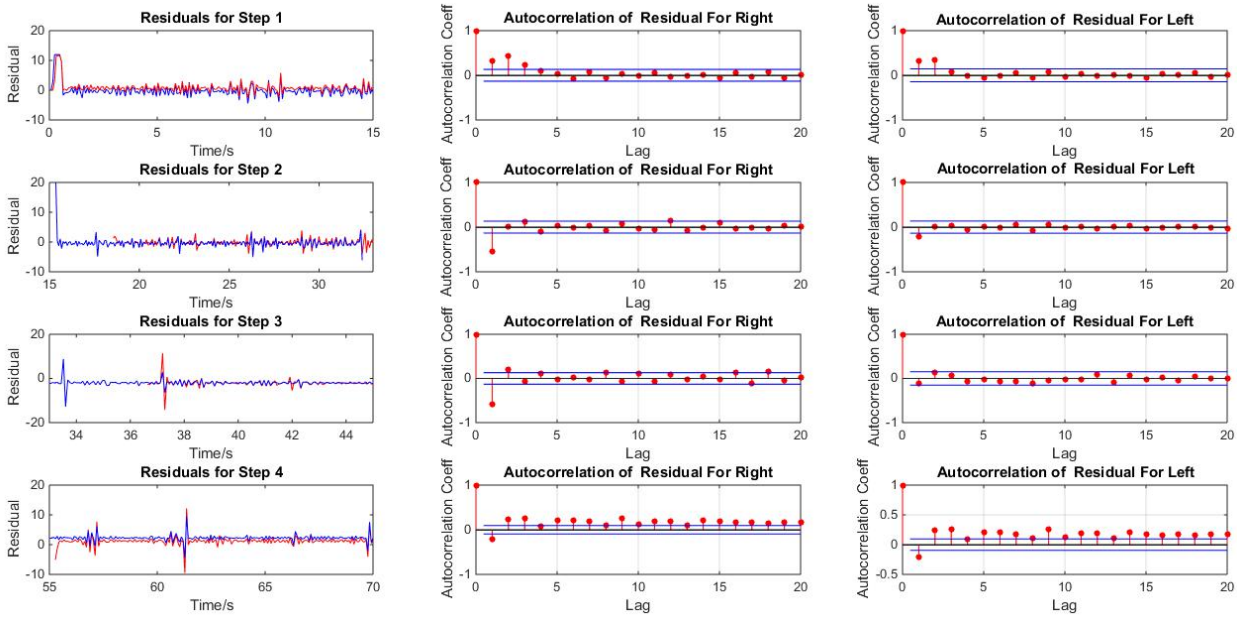


Figure 5 The residuals and its autocorrelation results

Task 5 PID controller for DaNI actuators using Labview

The digital PID controller is in the form of

$$u_n = K_p\left[e_n + \frac{1}{T_i}\left(I_{n-1} + e_n T\right) + T_d\left(\frac{e_n - e_{n-1}}{T}\right)\right]$$

$$= K_p\left[e_n\left(1 + \frac{T_d}{T} + \frac{T}{T_i}\right) - e_{n-1}\left(\frac{T_d}{T}\right) + \frac{I_{n-1}}{T_i}\right]$$

Where Kp, Ti, Td is proportional gain, integral time and derivative time respectively. The T is the sampling time.

There are some basic principles to tune the PID controller. The Proportional gain kp can determine the speed of the response. The large kp can lead to a fast growth as well as large overshoot and long settling time. The integral term is mainly used to eliminate the steady state error. It can also have a similar effect as kp. The smaller the integral time Ti is, the stronger the integral action is. The derivative term acts as a prediction. It can reduce the oscillation. However too much derivative term may cause the amplify of noise. The larger the Td is ,the stronger the prediction action is.

For this case, since the value of input is restricted within [-1,1], we decide not to use the Z-N method. Because the input is very likely to saturate. As a result, the specific value of kp to stimulate desired constant oscillatory response is hard to decide.

Instead, we use an very intuitive trail and error method to update the coefficients. Moreover, we record the error, the accumulated integral value and the derivative value and use these data to help us understand the system's response. We consider both the dynamic response and the specific value to make a decision. For example, if the system suffer from too slow response, we will see the relative proportion of both the proportional and integral term in the total input u. And synthesis the value of error and integral term to determine the scale to change. By this method, we may avoid overreaction and can specific what is the cause of the bad performance.

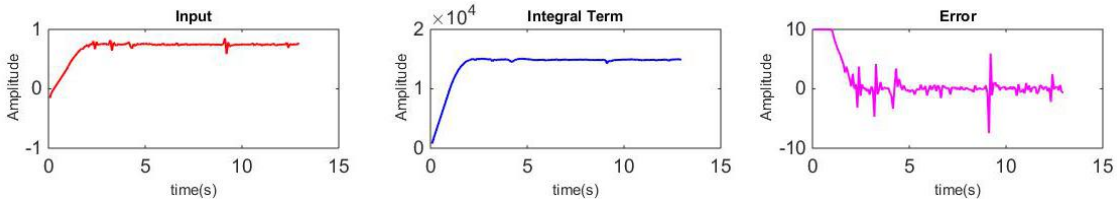


Figure 6 The input, integral error and tracking error during the PID

To begin with, we test the system with some inputs to determine the required steady state input. We find that to keep a angular velocity of 10 rad/s, the input should be around 0.74, which is very closed to the saturation value,i.e. 1. So the reaction of the controller cannot be too violent. With this intuitions ,we firstly set the derivative time to zero, the integral time is set to a large number, namely 3 times the sampling time(2.5x80=200). Then we test a series of kp to determine the boundary value of the saturation, which can be considered as the maximum value we can use for this integral time. Then tune down the kp little by little to eliminate overshoot. As far as my consideration, I prefer the slow but smooth response than the oscillatory response for the motor. Because the oscillation may cause the fatigue to the actuators and reduce the lifetime of usage.

Thus, we finally set a kp value of 0.01 when the Ti is 200. However the system take a long time to recover the steady state, which means the integral action is too weak. Thus we decrease the integral time. It should be noticed that the integral error is of a relative large value, the change of Ti may easily cause the saturate. As a result , we slightly decrease the kp to avoid the saturation. The final choice of kp is 0.006 and the Ti is 120. Then we add some derivative term to make the response more smooth, the final choice is 100. So our final parameters for PID controller is kp= 0.006, Ti=120, Td=100. The response is a little bit slow. The rise time with reference equal to 10 is about 100 ms. However there is no overshoot and the settling time is very close to the rise time. The dynamics is relatively smooth as my expectation.

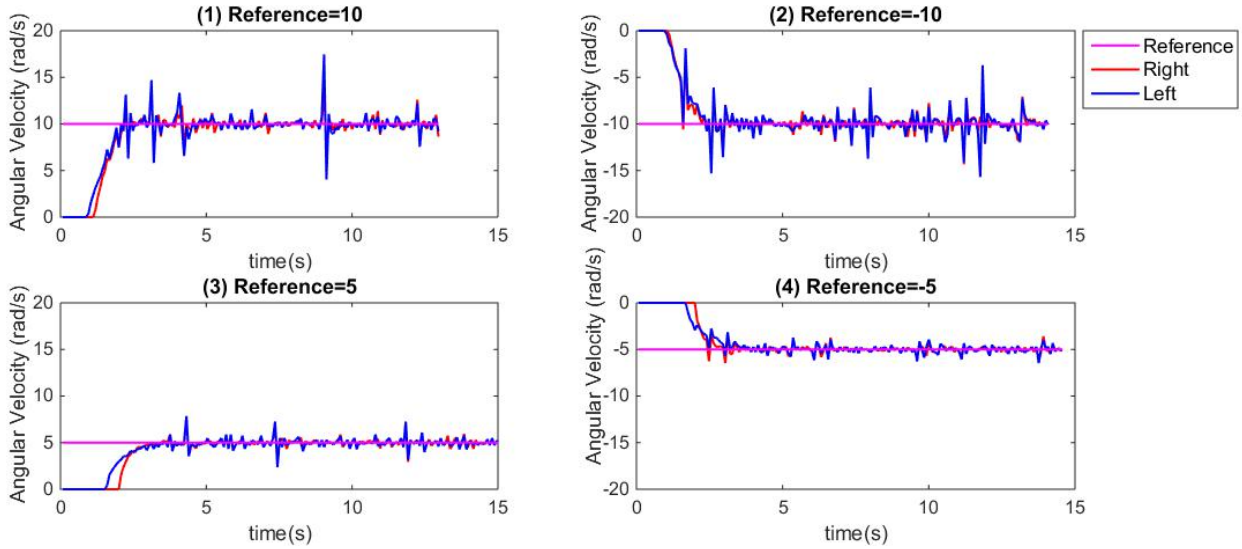


Figure 7 The tracking performance of the PID controller

Task 6 GPC for DaNI actuators using Labview

According to the results of system identification, the system can be modeled in the form of z-transform

$$a_L(z)\cdot y_L(k) = b_L(z)\cdot u_L(k) + \varepsilon$$
$$a_L(z) = 1 - 0.2767\cdot z^{-1} + 0.0033\cdot z^{-2}, b_L(z) = 2.7981\cdot z^{-1} + 9.1688\cdot z^{-2}$$
$$a_R(z)\cdot y_R(k) = b_R(z)\cdot u_R(k) + \varepsilon$$
$$a_R(z) = 1 - 0.3059\cdot z^{-1} + 0.0411\cdot z^{-2}, b_R(z) = 3.1468\cdot z^{-1} + 8.4937\cdot z^{-2}$$

Where epsilon is zero-mean white noise.

Thus the CARIMA model,which computing Δu to cancel the effect of noise, can be derived as

$$[a(z)\Delta]y_k = b(z)[\Delta u_k]$$
$$A(z) = a(z)\Delta$$
$$A(z)y_k = b(z)[\Delta u_k]$$

Then the H P Q matrix can be computed through some simple but messy ‘for’ loops in labview. Specifically, initialize the Ca, Cb, Ha, Hb matrix with the correct size. Next, using two embedded for loops to assembly each of these matrix. Finally compute the H, P ,Q matrix using the above results.

After obtaining the model, the next step is minimizing the cost function:

$$J = \left(\underset{\rightarrow}{\mathrm{r}}_{k+1} - \underset{\rightarrow}{y}_{k+1} \right)^T \left(\underset{\rightarrow}{\mathrm{r}}_{k+1} - \underset{\rightarrow}{y}_{k+1} \right) + \lambda \underset{\rightarrow}{\Delta \mathrm{u}}_k^T \underset{\rightarrow}{\Delta \mathrm{u}}_k$$

Computing the partial derivative of J with respect to the Δu_k. The theoretical solution can be computed as

$$\Delta \mathrm{u}_k = E(H^T H + \lambda I_p)^{-1} H^T \left(\underset{\rightarrow}{r}_{k+1} - P \underset{\leftarrow}{\Delta \mathrm{u}}_{k-1} - Q \underset{\leftarrow}{\mathrm{y}}_k \right)$$

$$u_k = u_{k-1} + \Delta \mathrm{u}_k$$

The set points are set as the requirement. I use some arrays and 'shift registers' to memorize the past Δu, past y and the future r. Then convert the arrays into matrix and compute the input u using H, P, Q matrix. A 'formula node' is used to realize the 'if' function in labview. Finally, the input can be recorded and implemented.

After getting everything ready, we begin to tune the parameters. There are two parameters in the GPC method, namely the horizon and lamda. To ensure the prediction of future change, the horizon have to be selected as larger than the settling time. The lamda is a weighting parameter for the input action and the tracking behaviour. Specifically, the larger lamda may restrict the input and result in smooth but slow response. In the other hand, the smaller lamda can lead to fast but fluctuating and energy-wasting response.

In the hardware, we test four groups of combination of lamda and horizon. The results are shown below. The overall tracking performance is acceptable. There are some delays in the response due to the static friction and emf of the motor. Increase the lamda from 1500 to 2000, the rise time is obviously slower. The large deviation from the steady state may due to noise. Increase horizon from 3 to 4, the prediction action is strength. The responses to the set point shift a little bit forward. Considering of the computation effort and the resulting response ,the appropriate horizon can be 3 and the lamda can be 1500.

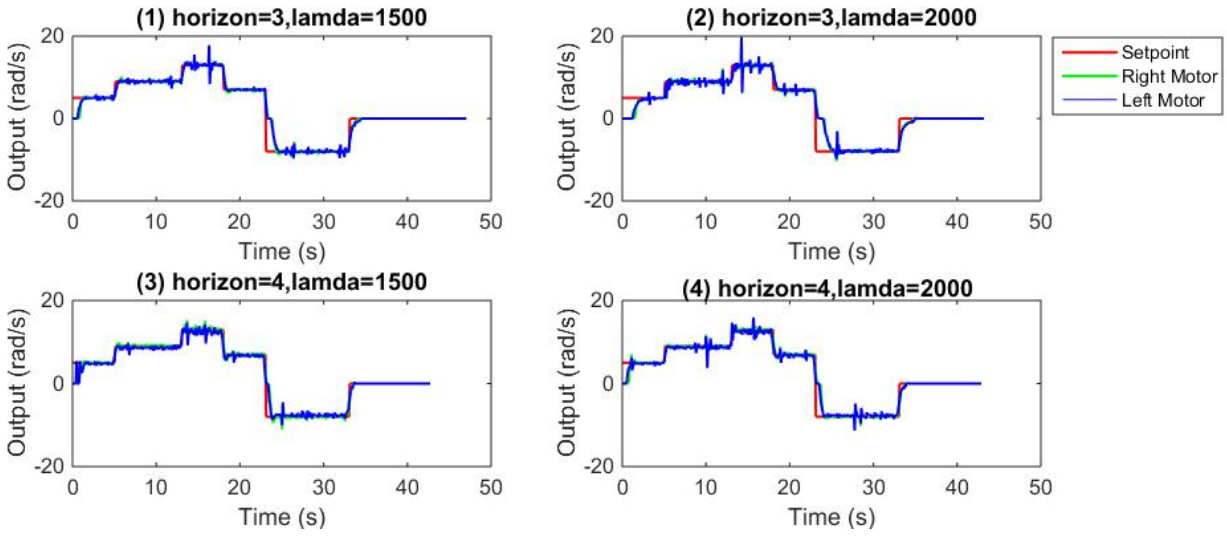


Figure 8 The tracking performance of GPC with different lamda and horizon