Digital Speed Governor for a Compressed Air Generator Case Study EE4002

Andrew Cooke
117379906

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

1. Introduction

The objective of this case study is to identify a model for a compressed air generator at the specified operating point and to design a suitable digital master controller to control the generator speed about this operating point. The (simplified) block diagram of the system is shown in Figure 1.

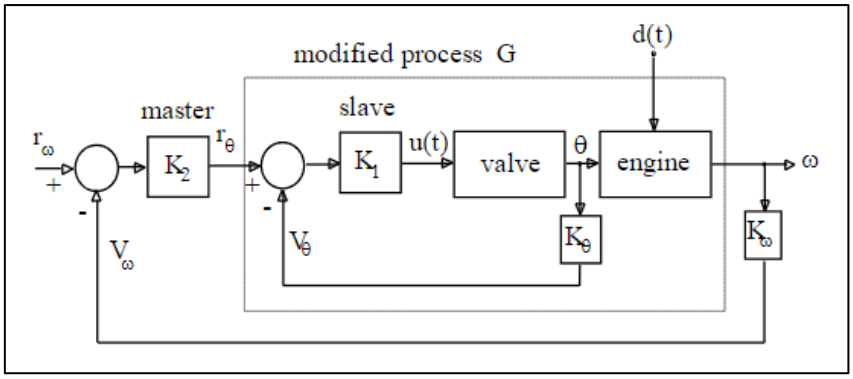


Figure 1: Block diagram of compressed air generator.

The compressed air generator is an example of a cascade control system, where the inner loop controls the valve position which feeds air to the generator and causes a change in speed. The outer loop controls the engine speed by controlling the valve position in the inner loop. The inner loop is already under control using a simple proportional controller in this case study.

The engine speed in volts is available from the sensor K_ω. The operating point of the generator is 28 rad/s, corresponding to a voltage of 1.4 V from the sensor. The disturbance $d(t)$ is caused by the load current, which is assumed to be independent of the engine speed. The load current is measured directly and is available from a sensor. The valve reference position $r_\theta(t)$ is initially left open. It was necessary to use these measurements to identify a discrete time model for the generator at the operating point, and hence use this model to design a digital master controller for the outer loop of the system which would control the engine speed.

2. Identifying the Reference Voltage Operating Point

For the first stage of the project it was necessary to identify the valve position reference voltage $r_\theta(t)$ that would provide an average generator speed of 28 rad/s (or 1.4 V, as measured by the sensor), which is the specified operating point.

The data from the sensor was sampled at the specified sampling period of 0.5 s. An anti-aliasing filter was applied before the data was sampled. The purpose of the filter was to prevent high frequency components in the generator from aliasing into the frequency band of interest, i.e. the frequency band of the generator outputs being monitored. According to the Shannon-Nyquist sampling theorem, the sampling frequency (2 Hz, or 12.57 rad/s in this case) should be twice the maximum frequency component of the signal being sampled. Thus, a first order low pass filter with corner frequency of 6 rad/s (approximately half the sampling frequency) was used to attenuate any high frequency noise components that may alias into the outputs of the generator. This anti-aliasing filter was applied to every sensor in the system before the data was sampled. The transfer function for the anti-aliasing filter is shown in Equation 1.

$$F(s) = \frac{6}{s+6} \tag{1}$$

A number of step inputs of varying sizes were then applied to the generator, and the engine speed (in volts) was plotted using the sampled sensor output. The engine speed for various step sizes is shown in Figure 2.

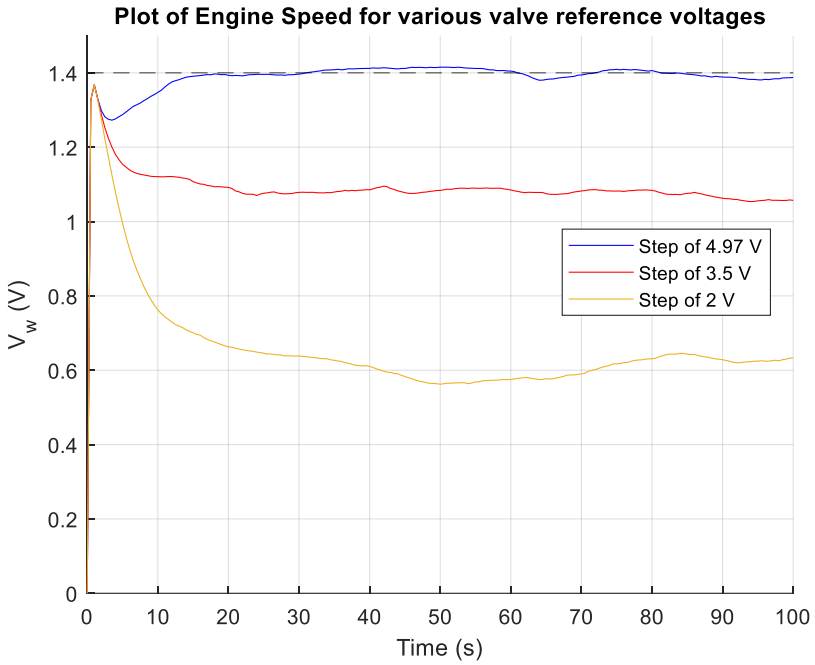


Figure 2: Engine speed V_ω (V) for various step sizes.

The dashed line at 1.4 V shows the specified engine speed operating point. It's clear from the plot that the step of 4.97 V in the valve position reference voltage $r_\theta(t)$ yields the desired average engine speed.

3. Identifying a Discrete Time Model of the Generator

The next stage of the case study involved identifying a discrete time model of the generator, with the inner loop (controlling the valve position) closed and the outer loop (controlling the engine or generator speed) open. This required the collection of experimental data from the engine inputs and outputs. The Least Squares algorithm was then applied to this data in order to identify the discrete time model. The first step in the process required the generation of a suitable excitation signal.

3.1 Excitation Signal and Data Collection

The choice of excitation signal is extremely important in system identification, as it is essential that the data you collect contains relevant information about the system dynamics – the system must be excited by all the frequencies in its operating range. As the sampling frequency was 2 Hz (giving a corner frequency of 1 Hz for the anti-aliasing filter), it was necessary to use an excitation signal that is rich in frequencies in the range of 0 to 1 Hz. A pseudo-random binary sequence (PRBS) was chosen for this task.

The PRBS signal was generated as an array of 1s and 0s using MATLAB's random integer generation function. The array had 401 elements, corresponding to a value for every timestamp between 0 and 200 s (inclusive) in steps of 0.5 s. This ensured that the smallest period of a given pulse (toggle from 1 to a 0 or vice versa) was 1 s. The PRBS was added to a step of 4.97 V before being applied to r_θ, the input of the generator. This ensured that the PRBS would be exciting the generator about its operating point (i.e. a speed of 28 rad/s). The Fast Fourier Transform (FFT) of the PRBS signal (without the 4.97 V offset) is shown in Figure 3. It is clear from the FFT that the PRBS is rich in frequencies ranging from 0 to 1 Hz.

Three experiments were run using separate, distinct PRBS sequences. The input and output data $r_\theta(t)$ and $V_\omega(t)$ collected from these three experiments corresponded to the training data for the Least Squares algorithm, the validation data for model selection and the test data to test how well the chosen model generalizes to unseen data. The training data was generated using a PRBS of amplitude 0.5 V. The validation data was also generated with a separate PRBS of 0.5 V amplitude. Finally, the test data was generated using a PRBS of amplitude 1 V. As stated previously, the PRBS was offset by 4.97 V (the operating point) for each experiment.

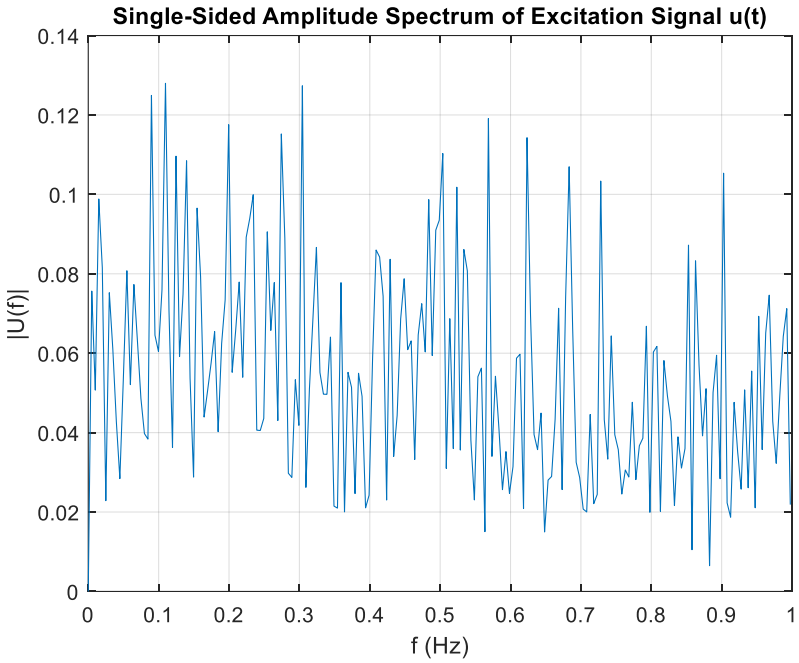


Figure 3: FFT showing the frequency content of the PRBS signal of 0.5 V amplitude.

3.2 The Least Squares Algorithm

The Least Squares Algorithm enables the discrete time transfer function for a Single Input, Single Output (SISO) system to be estimated using data collected from the system. The general transfer function for such a system is shown in Equation 2.

$$\frac{Y(z)}{U(Z)} = G(z) = \frac{z^{-d}(b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m})}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_n z^{-n}} \quad (2)$$

The difference equation of a candidate model for such a system can be written as shown in Equation 3, where $\widehat{a_i}$ and $\widehat{b_j}$ are estimates of the transfer function coefficients.

$$\hat{y}(k+1) = \widehat{a_1} y(k) + \cdots + \widehat{a_n} y(k-n+1) + \widehat{b_1} u(k-d) + \cdots + \widehat{b_m} u(k-d-m+1) \quad (3)$$

Obviously the candidate model is not valid for k = 0, as it requires information from before the recording began. The equation is only valid after *n* data points have been recorded (as it requires *n* past output values to make a prediction). Subsequently, predictions can be made for the output of the system at any time after the initial *n* data points have been recorded. This can be represented in matrix form, as shown in Equation 4.

$$\begin{bmatrix} \hat{y}(m+d) \\ \vdots \\ \hat{y}(N-1) \end{bmatrix} = \begin{bmatrix} y(m+d-1) & \dots & y(m+d-n) \\ \vdots & \vdots & \vdots \\ y(N-2) & \dots & y(N-n-1) \end{bmatrix} \begin{bmatrix} u(m-1) & \dots & u(0) \\ \vdots & \vdots & \vdots \\ u(N-d-2) & \dots & u(N-d-m-1) \end{bmatrix} \begin{bmatrix} \widehat{a_1} \\ \vdots \\ \widehat{b_m} \end{bmatrix} \quad (4)$$

This equation can be represented more concisely in the form shown in Equation 5, where $\underline{\widehat{Y_N}}$ is the column vector of N predictions made by the model, Φ is the regressor matrix containing the ordered data required for each prediction, and $\underline{\hat{\theta}}$ is the column vector of the estimated model parameters ($\widehat{a_1} \dots \widehat{a_n}, \widehat{b_1} \dots \widehat{b_m}$) corresponding to the transfer function coefficients.

$$\underline{\widehat{Y_N}} = \Phi \underline{\hat{\theta}} \quad (5)$$

In order to identify a suitable model of the compressed air generator system and hence to make accurate predictions of its output $V_\omega(t)$, two things must be ensured. The data contained in the regressor matrix Φ used to estimate the model parameters must be rich, containing information about the system dynamics over the entire range of operation, and the most accurate values of *n* and *m* must be identified for the model parameter vector $\underline{\hat{\theta}}$. Accurate values of *n* and *m* are the values that minimise the function shown in Equation 6, known as

the Least Squares cost function, where $e^2(i)$ is the squared error between the actual measured output and the predicted value i.e. $e^2(i) = \left(y(i) - \hat{y}(i)\right)^2$.

$$J = \sum_{i=m+d}^{N-1} e^2(i) \tag{6}$$

The values of the model parameters $\widehat{a_1}$ ... $\widehat{a_n}, \widehat{b_1}$ ... $\widehat{b_m}$ that minimise the Least Squares cost function can be computed using the closed form expression shown in Equation 7, where $\underline{Y_N}$ is a column vector of output values measured after the first *n* data points.

$$\underline{\hat{\theta}_{LS}} = (\Phi^T \Phi)^{-1} \Phi^T \underline{Y_N} \tag{7}$$

Thus, the minimised cost for various values of *n* and *m* can be compared in order to determine the most accurate model.

3.3 Estimating and Selecting Model Parameters

The PRBS has a rich enough frequency content in the range of 0 to 1 Hz that most of the system dynamics should appear in the output data $V_\omega(t)$ collected from the three experiments. This allows us to apply the Least Squares algorithm, with $u(kT) = r_\theta(kT)$ and $y(kT) = V_\omega(kT)$.

The training data that was collected from the first experiment was detrended and was then used to calculate $\underline{\hat{\theta}_{LS}}$ (the values of $\underline{\hat{\theta}}$ that minimise the cost function) for various values of *n* and *m*. These parameters were then used to make predictions on the training data (Equation 5) from which the cost for each choice of *n* and *m* could be computed using Equation 6. The costs (in V²) are shown in Table 1.

(n,m)	1	2	3	4	5	6	7	8
1	0.3690	NA	NA	NA	NA	NA	NA	NA
2	0.0054	0.0054	NA	NA	NA	NA	NA	NA
3	0.0018	0.0009	0.0009	NA	NA	NA	NA	NA
4	0.0018	0.0018	0.0009	0.0008	NA	NA	NA	NA
5	0.0017	0.0017	0.0017	0.0008	0.0008	NA	NA	NA
6	0.0017	0.0017	0.0016	0.0016	0.0007	0.0006	NA	NA
7	0.0017	0.0017	0.0016	0.0016	0.0016	0.0007	0.0006	NA
8	0.0017	0.0017	0.0017	0.0016	0.0016	0.0016	0.0007	0.0006

Table 1

The lowest cost in each row (i.e. for each choice of *n*) occurs for *n*=*m*. The lowest costs occur for the highest values of *n* and *m* (i.e. *n*=*m*=6, 7, 8). While it may be tempting to immediately select one of these values for both *n* and *m*, there is a reason that a validation and test data set were also collected. Computing the costs for the validation data using the same $\underline{\hat{\theta}_{LS}}$ parameters as used previously yields the costs shown in Table 2.

(n,m)	1	2	3	4	5	6	7	8
1	477.593	NA	NA	NA	NA	NA	NA	NA
2	0.01182	0.01895	NA	NA	NA	NA	NA	NA
3	0.09625	0.00800	0.00176	NA	NA	NA	NA	NA
4	0.10945	0.10445	0.01279	0.00372	NA	NA	NA	NA
5	0.1004	0.11204	0.11095	0.01574	0.00592	NA	NA	NA
6	0.07797	0.09210	0.10298	0.10039	0.01730	0.00699	NA	NA
7	0.09062	0.09601	0.11129	0.11820	0.11736	0.02289	0.01010	NA
8	0.09106	0.10701	0.11542	0.12783	0.12664	0.12674	0.02336	0.01069

Table 2

The costs shown in Table 2 are significantly larger on average for every choice of *n* and *m*. Computing the cost on the training data, i.e. the same data used to calculate $\underline{\hat{\theta}_{LS}}$, gives an over-optimistic estimate of the actual

cost. The model parameters $\underline{\hat{\theta}}_{LS}$ for each choice of *n* and *m* were tailored to minimise the cost on the training data set using Equation 7. As a result, the cost is about two orders of magnitude greater on average over the validation set for each choice of *n* and *m*. Furthermore, from Table 2 it can be seen that the cost is no longer at its lowest for the largest choices of *n* and *m* (e.g. 6, 7, and 8). Such large values of *n* and *m* actually cause the model parameters to overfit the training data set, meaning the model can make predictions with incredible accuracy for data it has already seen, but is unable to generalize to new, unseen data. Interpreting the difference in costs is much easier when visualized, hence the cost on the training and validation sets for *n*=*m* is shown in Figure 4. Note that the cost for *n*=*m*=1 was cut off from the plot as the value was so large for both data sets that it squashed the curves into barely visible straight lines.

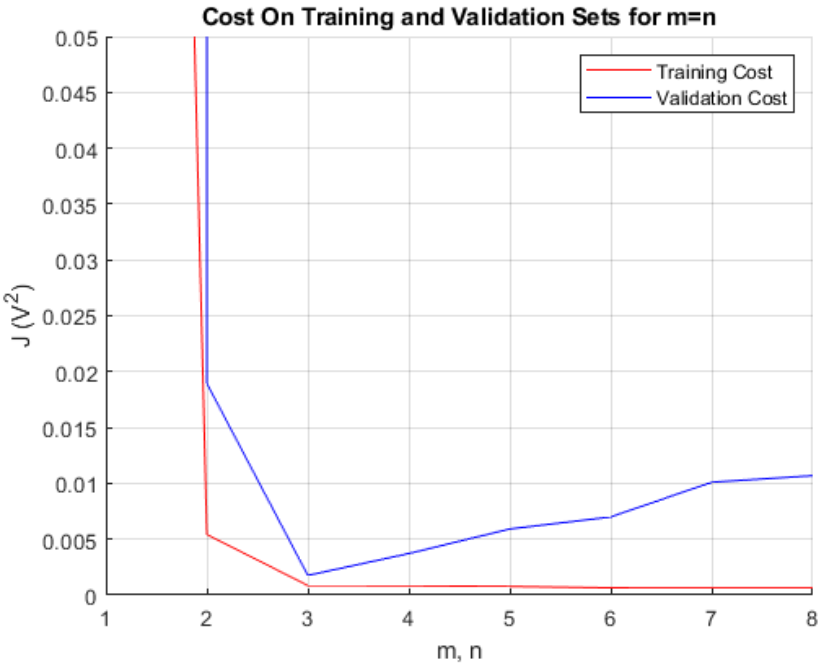


Figure 4: Costs computed on the training and validation sets for n=m.

It's clear from Figure 4 that the lowest cost on the validation set occurs for *n*=*m*=3 - from Table 2, the exact cost is 0.00176 V². For values larger than *n*=*m*=3, it can be seen in Figure 4 that the validation cost begins to increase, while the training cost continues to decrease – i.e. the model begins to overfit the training data. Thus, it's clear that *n*=*m*=3 give the most accurate estimate of the actual system parameters. $\underline{\hat{\theta}}_{LS}$ for this choice of *n* and *m* is shown here.

$$\underline{\hat{\theta}}_{LS} = [1.7156 \quad -0.7696 \quad 0.0369 \quad 0.0007 \quad 0.003 \quad 0.0008]^T$$

The last set of data collected, the test set, can be used to obtain a final estimate of the generalization error of the selected model (i.e. the selected choice of *n* and *m*). As mentioned previously, the data in the test set was collected from the system using a PRBS excitation signal with an amplitude of 1 V. Computing the cost on the test set yielded a final cost of 0.00223 V². Thus, we can be reasonably sure that out model will generalize well to new, unseen data.

3.4 Identifying the Discrete Time Transfer Function

The selected Least Squares model parameters can be used to create a discrete time transfer function of the form shown in Equation 2 for the compressed air generator system. Note that for this system, the z^{-d} term is 1, i.e. *d* = 0. The estimated transfer function of the system is shown in Equation 8.

$$G(z) = \frac{0.0007386z^{-1} + 0.003025z^{-2} + 0.000751z^{-3}}{1 - 1.716z^{-1} - 07696z^{-2} - 0.0369z^{-3}} \tag{8}$$

The pole-zero map (i.e. the plot of the poles and zeros of the system on the z-plane) of the transfer function is shown in Figure 5. It can be seen from the pole-zero map that all of the system poles lie in the unit circle, meaning the system is stable. The poles and zeros also lie on the real axis. The exact location of the poles and

zeros of the transfer function are shown in Table 3. Once the system transfer function was obtained, a digital master controller could then be designed for the outer loop of the compressed air generator that would control the speed $V_\omega(t)$ of the engine/generator.

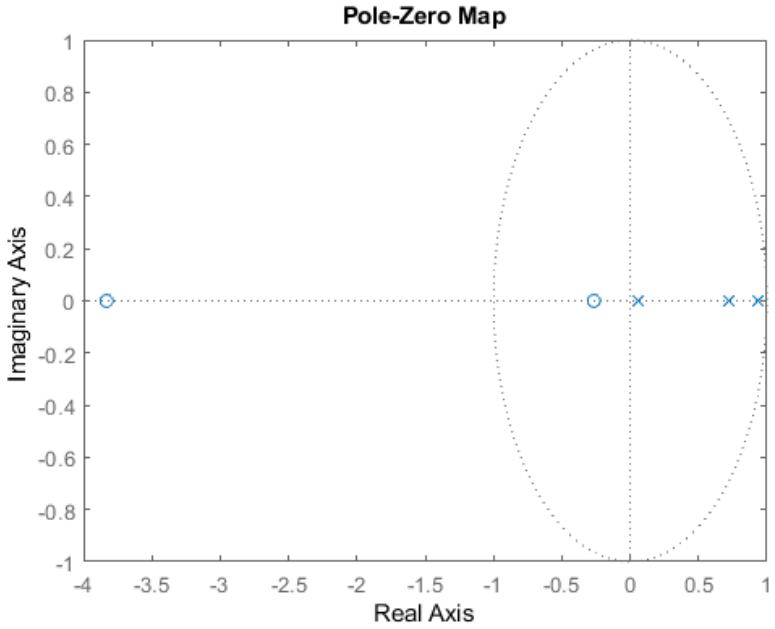


Figure 5: Pole-zero map of the estimated system transfer function.

Pole 1	Pole 2	Pole 3	Zero 1	Zero 2
0.9339+0j	0.7274+0j	0.0543+0j	-3.8303+0j	-0.2655+0j

Table 3

4. Digital Master Controller Design and Testing

A PID controller was selected for the task of controlling the engine speed in the outer loop of the cascade control system, as its proportional, integral, and derivative terms combined allow robust control of the system output. In particular, the integral term allows the elimination of steady state error in the system output and the derivative term enables the controller to take corrective action proportional to the rate of change of the error. As the system transfer function was already identified, the controller could be easily designed using root locus. The only design specification for the controller was that it must have a damping factor greater than 0.5.

4.1 Root Locus Design

As stated previously, the root locus method was used to design the digital PID master controller. The controller was designed to have a closed loop damping of approximately 0.8 and a frequency of 1.1 rad/s. Thus, the DC gain and zero of the controller had to be calculated such that the root locus would pass through the point $z = 0.6 + 0.2j$ to ensure the controller met the design criteria. An extra pole at the origin had to be introduced to the normal Z-domain transfer function of the PID controller in order to make it a causal system (otherwise it would require future information to function). The derivation of the PID controller transfer function using root locus is shown in Appendix A. The PID controller transfer function (including the extra pole at the origin) is shown in Equation 9.

$$D(z) = \frac{67.5(z - 0.793)^2}{z(z - 1)} \tag{9}$$

The unit circle of the root locus plot of D(z)G(z) is shown in Figure 6, from which the controller poles and zeros can be seen. The point $z = 0.6 + 0.2j$ is annotated to show the root locus passing through it.

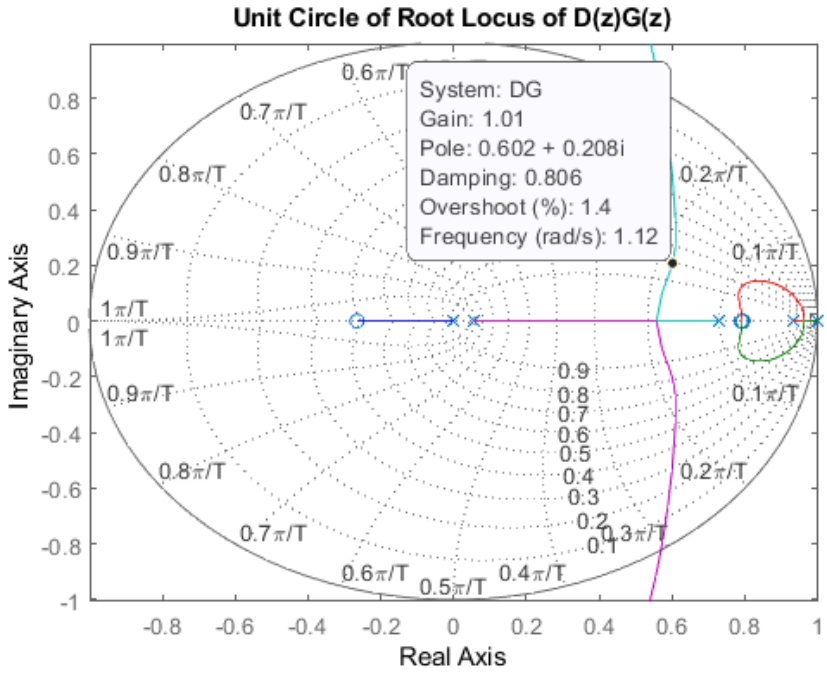


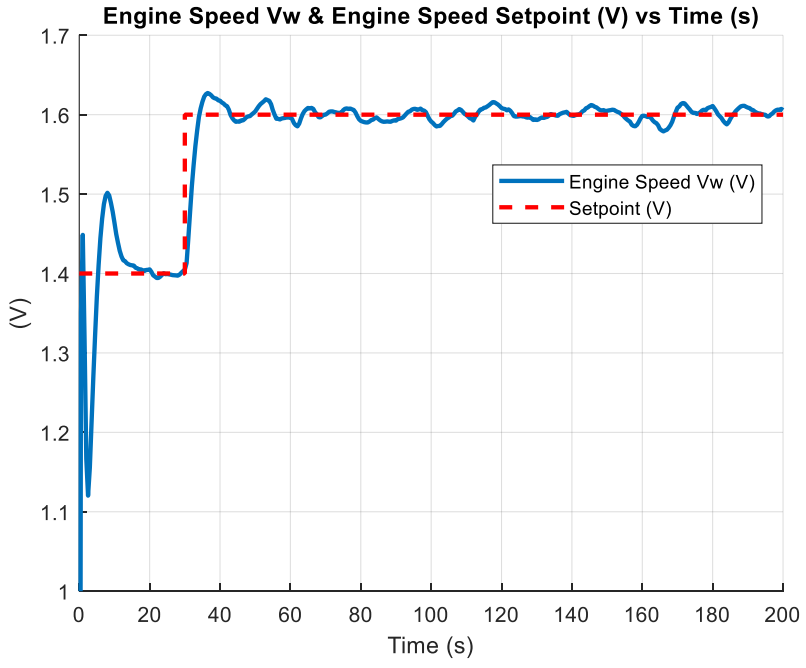
Figure 6: Unit circle of Root locus plot of D(z)G(z).

From the annotated root locus plot shown in Figure 6, it can be seen that the controller transfer function meets the design criteria.

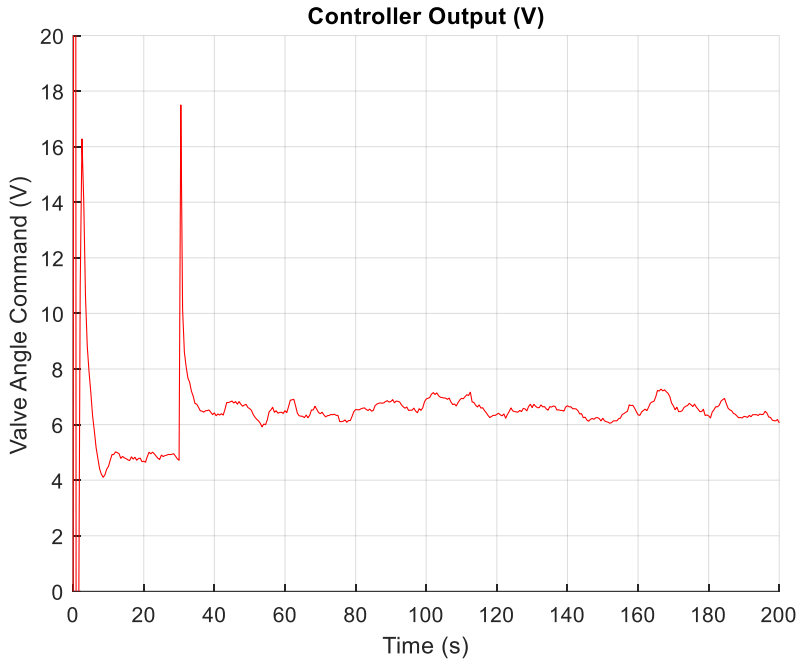
4.2 Testing the Digital Master Controller

The PID controller was implemented and the full Simulink diagram of the closed loop system can be seen in Appendix B. The controller was tested for two different set points.

A step input with an initial value of 1.4 V (the generator operating point) was applied first. A step to 1.6 V then occurs after 30 s, as this gives time for the system to settle at the operating point of 1.4 V. The engine speed and the corresponding setpoint are shown in Figure 7(a). The controller output for this test is shown in Figure 7(b). Note that the y-axis has been limited to a range of 1 – 1.7 V in 7(a) and a range of 0 – 20 V in 7(b) so that the waveforms could be viewed more clearly.



(a)

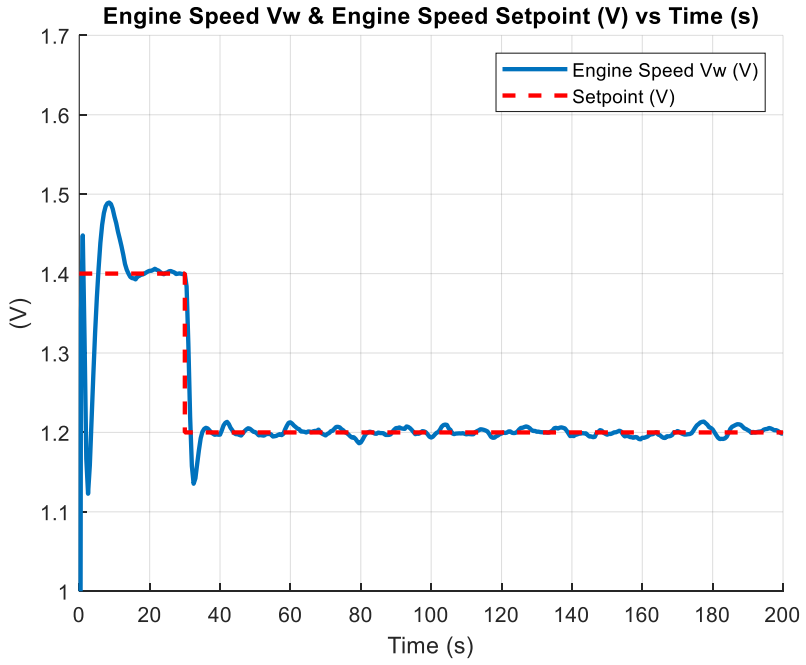


(b)

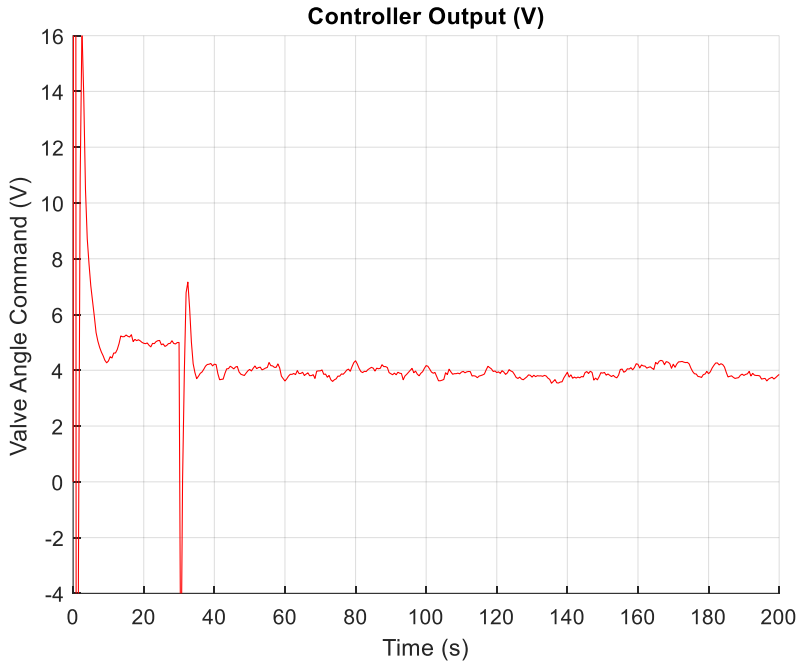
Figure 7: Engine speed and controller output for a step input from 1.4 V to 1.6 V.

The engine speed overshoots to a value of 1.628 V in Figure 7(a), which corresponds to the 1.4% peak overshoot specified by the annotated root locus plot in Figure 6. It's important to note that the engine speed does not settle completely at the setpoint of 1.6 V – there is some fluctuation in the engine speed about the 1.6 V point due to the load current in the generator acting as a continuously changing disturbance.

A second step test was then carried out. For this test, the initial value of the step was set to 1.4 V, and the final value was set to 1.2 V, with the step (down) occurring after 30 s once again. A plot of the engine speed and corresponding setpoint is shown in Figure 8(a), as well the corresponding controller output in 8(b). Note that a different scale was used in Figure 8(b) to Figure 7(b).



(a)



(b)

Figure 8: Engine speed and controller output for a step input from 1.4 V to 1.2 V.

The peak overshoot performance degraded for the step down to 1.2 V – the engine speed overshot to approximately 1.14 V, corresponding to an overshoot of 5%. This is a significantly larger overshoot than the 1.4% overshoot specified for the controller. It's possible that the disturbance caused by the load current was at a particularly low value during the step which may have caused this large overshoot. It's also possible that this overshoot was caused by simply pushing the system too far from its operating point – it's important to remember that the controller was designed for the operating point at 1.4 V.

It's clear from both of these step tests that while the controller manages to bring the system close to the chosen setpoint, the load current prevents the system from truly settling at the desired setpoint - instead causing the engine speed to fluctuate around the setpoint. In order to combat this issue, a feed-forward controller could be designed for disturbance rejection and implemented in the system.

5. Disturbance Rejection with Feed-Forward Control

In order to design a feed forward controller for disturbance rejection, it was necessary to repeat a number of steps, this time using the load current $d(t)$ as the input, with $V_\omega(t)$ again serving as the output.

5.1 Estimating the Disturbance Model of the System

It was once again necessary to run a number of experiments in order to collect data for the Least Squares algorithm so that the disturbance model of the compressed air generator system could be estimated. The valve position reference voltage $r_\theta(t)$ was offset to 4.97 V, to ensure the generator was at its operating point of 1.4 V. The load current $d(t)$ and the engine speed $V_\omega(t)$ were sampled at a sampling period of 0.5 s. An anti-aliasing filter with a corner frequency of 6 rad/s was again applied before the data was sampled. The load current source is independent of the compressed air generator system, as stated previously. Thus, no external excitation signal was applied. Two experiments were run to collect data for a training and a validation set. $\underline{\hat{\theta}}_{LS}$ was calculated for various values of *n* and *m* and the cost was then computed on the validation set for these values of *n* and *m*. The mean from the training data was used to detrend both the training and validation data sets. The computed costs are shown in Appendix C.

While the cost decreased for larger values of *n* and *m*, the cost was still reasonably low for *n*=*m*=1. This choice of *n* and *m* greatly simplifies the design of the feed-forward controller and so the simple first order model was selected. The resulting discrete time disturbance transfer function is shown in Equation 10.

$$G_D(z) = \frac{-0.005582z^{-1}}{1 - 0.1191z^{-1}} \tag{10}$$

5.2 Designing the Feed-Forward Controller

The disturbance transfer function can be combined with the inverse of the compressed air generator system transfer function $G(z)$ to design the feed-forward controller transfer function $C_D(z)$, as shown in Equation 11 [1].

$$C_D(z) = \frac{-G_D(z)}{G(z)} \tag{11}$$

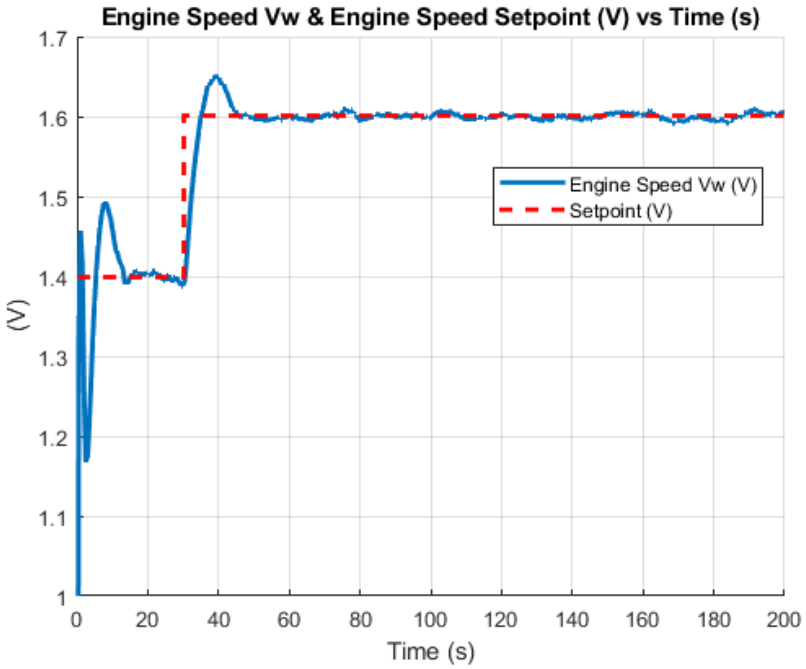
However, $G(z)$ is a non-minimum phase system, meaning that while $G(z)$ is causal and stable, its inverse is not. This can be easily verified using Figure 5: the zero at $-3.8303 + 0j$ will become a pole when $G(z)$ is inverted, meaning the system will become unstable. In such a scenario, it is possible to define $G^{inv}(z)$ as the invertible part of the transfer function $G(z)$ [2], provided the DC gain of the system transfer function is scaled by an amount equal to the distance from the removed zero to the test point $0.6 + 0.2j$ that was used while carrying out the root locus design of the PID controller. Finally, it is necessary to introduce a pole at the origin in order to make the feed forward controller transfer function $C_D(z)$ causal. The final feed forward controller transfer function is shown in Equation 12.

$$C_D(z) = \frac{-G_D(z)}{z \cdot G^{inv}(z)} = \frac{0.005582z^3 - 0.009576z^2 + 0.004295z - 0.000206}{0.003276z^3 + 0.0004795z^2 - 0.0001036z} \tag{12}$$

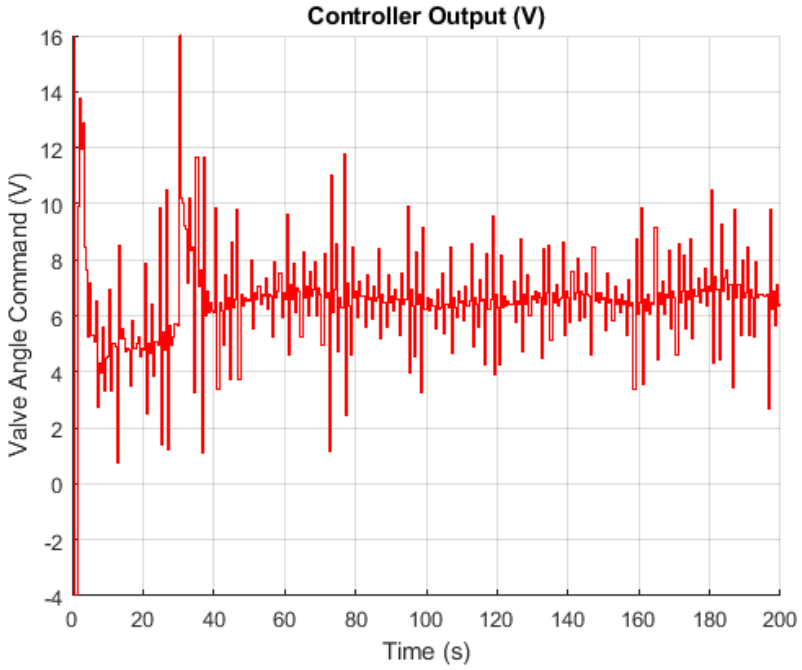
5.3 Testing the Feed-Forward Controller

The controller was implemented, and the full Simulink diagram can be seen in Appendix D.

A step test was then carried out on the overall system, with an initial value of 1.4 V and a step to 1.6 V after 30 s. The resulting engine speed and controller output (sum of the PID controller and the feed-forward controller outputs) for this test is shown in Figure 9.



(a)



(b)

Figure 9: Engine speed and controller output for a step input from 1.4 V to 1.6 V using the PID and feed-forward controller.

A number of things are immediately clear from the plots in Figure 9. The engine speed is a lot steadier and has a lot less fluctuation at both 1.4 V and 1.6 V compared to Figure 7(a) using just the PID controller. There doesn't appear to be any noticeable difference in the peak overshoot of the engine speed. However, there is a lot more rapid adjustments present in the controller output in Figure 9(b) compared to Figure 7(b). At its worst, the controller causes the valve reference position voltage to jump from 3 V to 12 V. These large jumps in the valve position will eventually wear out the valve actuators in the compressed air generator. While the steady state performance of the system is better using the feed-forward controller with the PID controller, this performance improvement must be weighed against the extra wear and tear that may occur in the system.

6. Conclusion

In this case study, the discrete time transfer function of the compressed air generator system at its operating point of 28 rad/s was identified using the Least Squares algorithm. The accuracy of the model was tested using validation and test data collected from experiments with the generator which was excited using a PRBS input signal. The estimated transfer function of the system was then used to design a digital master PID controller to control the engine speed – i.e. the outer loop of the cascade control system. This master controller was tested for various step sizes. Finally, a feed-forward controller was designed, implemented and tested for disturbance rejection, which improved the steady state performance of the system.

Further work on the feed-forward controller could be done by carrying out a more advanced controller design method. The method used in this case study is possibly too simplistic to be very effective – this can be seen by the presence of some slight fluctuations about the setpoint even after the implementation of the feed-forward controller.

References

[1] H. Zhong, L. Pao and R. de Callafon, "Feedforward control for disturbance rejection: Model matching and other methods," *2012 24th Chinese Control and Decision Conference (CCDC)*, Taiyuan, 2012, pp. 3528-3533, doi: 10.1109/CCDC.2012.6244565.

[2] J. Zhang. (2002). Internal Model Control [Online]. Available: https://bit.ly/3mchqIt

Appendix A

General form for a PID controller transfer function in the Z-domain with an extra pole at the origin to make the controller causal:

$$D(z) = \frac{K_c(z+a)^2}{z(z-1)}$$

Controller design specification is a closed loop damping of $\mathrm{Z} = 0.8$ and a frequency of $\omega_n = 1.1$ rad/s, meaning the root locus must pass through the point $z = 0.6 + 0.2j$.

Compressed air generator system transfer function (identified using Least Squares):

$$G(z) = \frac{0.0007386(z+3.8303)(z+0.2655)}{(z-0.9339)(z-0.7274)(z-0.0543)}$$

For any point $z = \zeta$ on the root locus:

$$\angle D(z)G(z)|_{z=\zeta} = 180°$$

$$|D(z)G(z)|_{z=\zeta} = 1$$

Thus, for the first criterion of the root locus:

$$\phi_{c1} + \phi_{c2} + \phi_1 + \phi_2 + \phi_3 - 2\theta_c - \theta_1 - \theta_2 = 180°$$

Where ϕ_{c1}, ϕ_{c2} and ϕ_1, ϕ_2, ϕ_3 are the angles with the real axis to the point $z = 0.6 + 0.2j$ of the controller and system poles respectively, θ_c and θ_1, θ_2 are the angles with the real axis to the point $z = 0.6 + 0.2j$ of the double controller zero and the system zeros respectively.

A sample calculation is shown here for ϕ_{c1}, the controller pole at $z = 0 + 0j$.

$$\phi_{c1} = \tan^{-1}\left(\frac{0.2}{0.6}\right) = 18.435°$$

This process was repeated for all the controller and system poles and zeros. The double zero of the controller was calculated to be $z = 0.793 + 0j$.

For the second criterion of the root locus:

$$\frac{K_c r_c^2 K r_1 r_2}{R_{c1} R_{c2} R_1 R_2 R_3} = 1$$

Where K_c and K are the DC gains of the controller and system transfer functions respectively, r_c and r_1, r_2 are the distances from the double controller zero and the system zeros to the point $z = 0.6 + 0.2j$ respectively and R_{c1}, R_{c2} and R_1, R_2, R_3 are the distances from the controller poles and the system poles to the point $z = 0.6 + 0.2j$ respectively.

A sample calculation is shown here for R_{c2}, the controller pole at $z = 1 + 0j$.

$$R_{c2} = |(0.6 + 0.2j) - 1| = \sqrt{(-0.4)^2 + 0.2^2} = 0.447$$

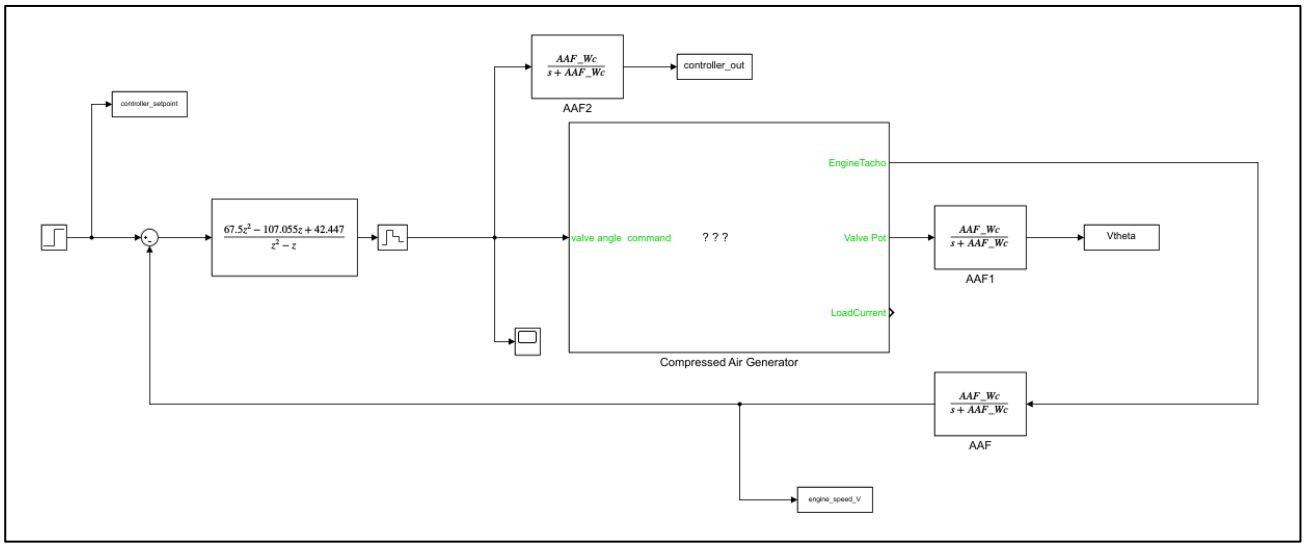
This process was repeated for all the controller and system poles and zeros. The DC gain of the controller K_c was calculated to be 67.5.

The final controller transfer function is shown here:

$$D(z) = \frac{67.5(z-0.793)^2}{z(z-1)}$$

Appendix B

PID controller implementation in Simulink model.



Appendix C

Cost on the validation set for the disturbance model of the compressed air generator system.

(n,m)	1	2	3	4	5	6	7	8
1	0.38523	NA	NA	NA	NA	NA	NA	NA
2	0.00935	0.009178	NA	NA	NA	NA	NA	NA
3	0.00404	0.0010	0.00076	NA	NA	NA	NA	NA
4	0.00391	0.00397	0.00104	0.00079	NA	NA	NA	NA
5	0.00392	0.00379	0.00381	0.00104	0.00078	NA	NA	NA
6	0.00405	0.00401	0.00378	0.00386	0.00100	0.00076	NA	NA
7	0.00404	0.00406	0.00402	0.00378	0.00389	0.00097	0.00072	NA
8	0.00404	0.00407	0.00410	0.00405	0.00382	0.00393	0.00097	0.00071

Appendix D

Feed-forward controller implementation in Simulink model.