

A Survey of Control Systems Applied to the Idle Control of an Automotive Engine

Jeremiah Mahler
jmahler@mail.csuchico.edu

CSU Chico
November 28, 2013

DRAFT

1 Engine Model

The engine model used here is based work by Butts and Sivashankar¹ which is derived from the work by Powell and Cook.² The engine configuration is a modern 4.6L V-8. To simplify analysis the linearized model is used as shown in Figure 1.

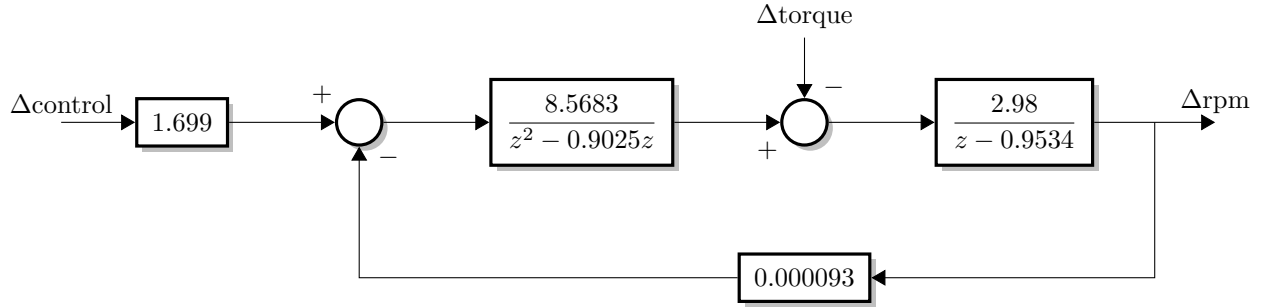


Figure 1: Linear engine model of a modern 4.6L V-8.

This model takes two inputs: a torque, and a idle control signal. When the torque is greater than zero it will oppose the rotation of the engine causing it to slow down. The idle control signal is some fraction of unity. This fraction corresponds to a pulse width modulated idle control valve which is at a minimum near zero and at a maximum near unity. Often the duty cycle range is in a range from 0% to 100% which corresponds to 0 to 1 (unity).

Because all the inputs and outputs are defined as deltas (Δ) this model cannot be used directly with typical control systems which use steady state values. It is possible convert these deltas to steady state equivalents. Figure 2 shows the transfer function to convert steady state values to delta values. Figure 3 shows the transfer function to delta values to steady state values.

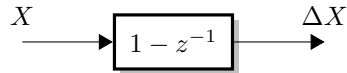


Figure 2: The Z transform used to accumulate the input and convert a steady state input to delta output. Its derivation is given in Appendix A.

¹K. Butts, N. Sivashankar, and J. Sun. "Feedforward and feedback design for engine idle speed control using l1 optimization". In: *American Control Conference, Proceedings of the 1995*. Vol. 4. 1995, 2587–2590 vol.4. DOI: 10.1109/ACC.1995.532315.

²B.K. Powell and J. A. Cook. "Nonlinear Low Frequency Phenomenological Engine Modeling and Analysis". In: *American Control Conference, 1987*. 1987, pp. 332–340.

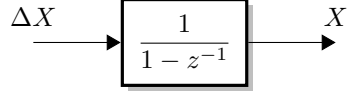


Figure 3: The Z transform used to convert delta input to a steady state output. Its derivation is given in Appendix B.

Typical control systems have an associated time step which can be varied and is crucial in determining the performance of the controller with regard to the Nyquist frequency. This model has no such constraint because one calculation step corresponds directly to one engine ignition event. The controller is effectively running at the same “clock speed” as the engine.

References

Butts, K., N. Sivashankar, and J. Sun. “Feedforward and feedback design for engine idle speed control using l1 optimization”. In: *American Control Conference, Proceedings of the 1995*. Vol. 4. 1995, 2587–2590 vol.4. DOI: 10.1109/ACC.1995.532315.

Octave community. *GNU/Octave*. 2012. URL: www.gnu.org/software/octave/.

Powell, B.K. and J. A. Cook. “Nonlinear Low Frequency Phenomenological Engine Modeling and Analysis”. In: *American Control Conference, 1987*. 1987, pp. 332–340.

A Steady State to Delta Transform Derivation

To accumulate a steady state input to produce a delta output a system can be constructed as shown in Figure 4. Its operation can be confirmed by trying some values. If all values are zero and then a 1 is input on u the output will become 1. On the next time step 1 will be output on v . Since q is zero r will be 1. If the input (u) remains 1 this will be subtracted from r to produce zero on the output (y).

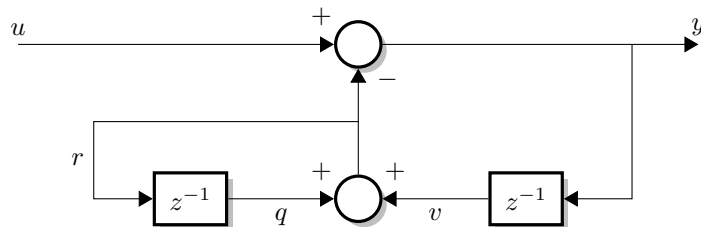


Figure 4: System to accumulate values to convert a steady state input to a delta output.

Figure 5 shows the response of this system given an arbitrary input. It can be seen that if the input is held constant the output (delta) returns to zero as expected.

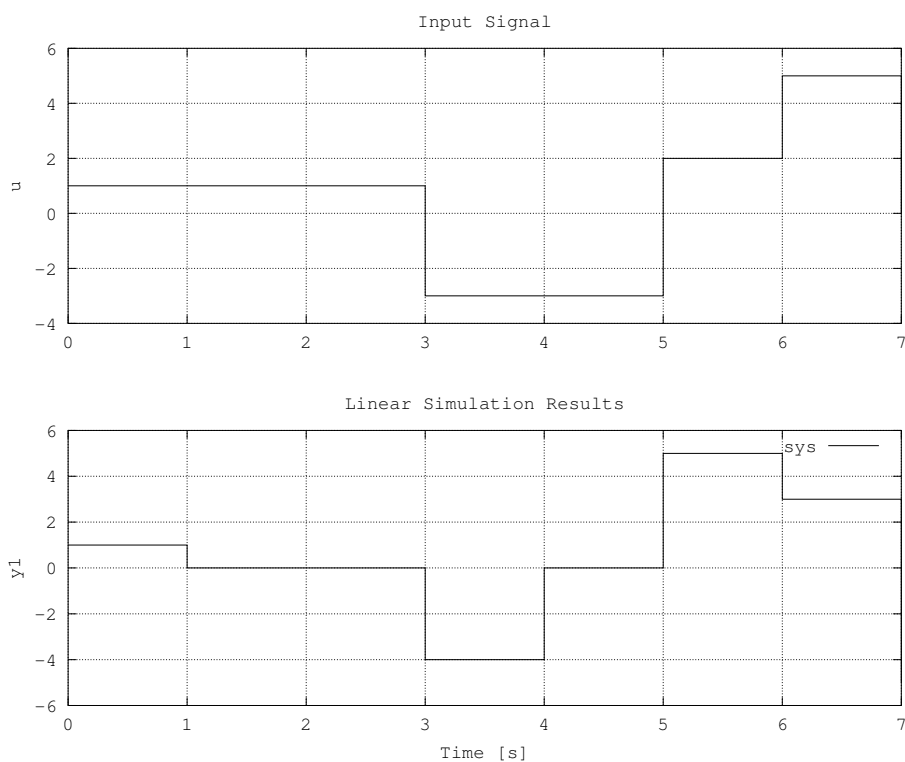


Figure 5: Response of full steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 1 and 2.

However this full system can be simplified to a single transfer function. Starting from the equations that define the system

$$r = q + v \quad (1)$$

$$v = y \cdot z^{-1} \quad (2)$$

$$q = r \cdot z^{-1} \quad (3)$$

$$y = u - r \quad (4)$$

these can be algebraically manipulated to find the effective transfer function of the entire system (y/u).

$$\begin{aligned} r &= rz^{-1} + yz^{-1} & (1, 2, 3) \\ r(1 - z^{-1}) &= yz^{-1} \\ r &= u - y & (4) \\ (u - y)(1 - z^{-1}) &= yz^{-1} \\ u - y - uz^{-1} + yz^{-1} &= yz^{-1} \\ u - y - uz^{-1} &= 0 \\ y &= u(1 - z^{-1}) \end{aligned}$$

$$\boxed{\frac{y}{u} = 1 - z^{-1}} \quad (5)$$

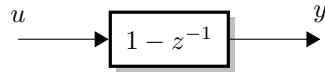


Figure 6: Simplified system to convert a steady state input in to a delta output.

It can be seen in Figure 7 that the simplified system behaves identically to the previous system (Figure 5).

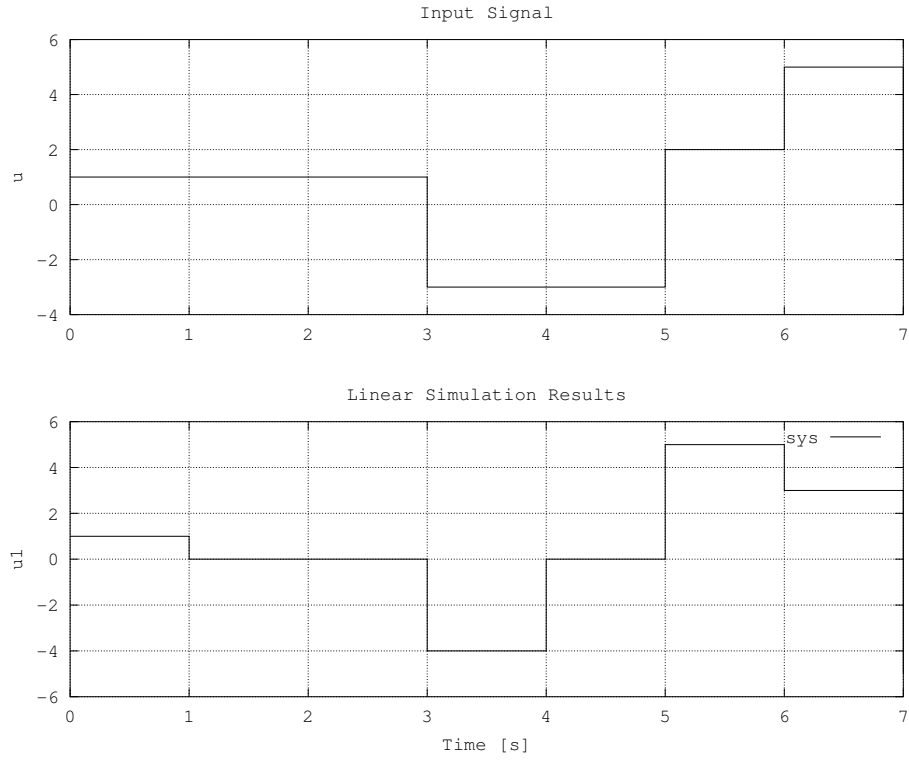


Figure 7: Response of simplified steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). Response is identical to the full system in Figure 5 as expected. The Matlab source code is given in Listing 3 and 4.

A.1 Matlab Source

The following code has been tested using Octave,³ an open source Matlab clone.

```
1 %
2 % cd1_init.m
3 %
4 % Convert a steady state input to a delta output.
5 %
6 % Full System.
7 %
8
9 T = 1; % time step
10
11 D1 = tf([1], [1 0], T, 'inname', 'y1', 'outname', 'v1');
12 D2 = tf([1], [1 0], T, 'inname', 'r1', 'outname', 'q1');
13 sum1 = sumblk('y1 = u1 - r1');
14 sum2 = sumblk('r1 = v1 + q1');
15 sys = connect(D1, D2, sum1, sum2, 'u1', 'y1');
```

Listing 1: Matlab code to initialize the full steady state to delta system.

```
1 %
2 % cd1_plot.m
3 %
4
5 clear;
6
7 cd1_init;
8
9 u = [1 1 1 -3 -3 2 5 0];
10 t = 0:(size(u,2)-1); % start at zero
11
12 figure;
13 subplot(2,1,1);
14 stairs(t,u);
15 grid on;
16 axis auto;
17 title('Input Signal');
18 ylabel('u');
19
20 subplot(2,1,2);
21 lsim(sys, u);
22 grid on;
23 axis auto;
24
25 print('cd1-plot.eps', '-deps');
```

Listing 2: Matlab code to plot the full steady state to delta system.

³Octave community. *GNU/Octave*. 2012. URL: www.gnu.org/software/octave/.

```

1  %
2  % cd2_init.m
3  %
4  % Convert a steady state input to a delta output.
5  %
6  % Simplified System.
7  %
8
9  T = 1; % time step
10
11 sys = tf([1 -1], [1 0], T, 'inname', 'y1', 'outname', 'u1');
```

Listing 3: Matlab code to initialize the simplified steady state to delta system.

```

1  %
2  % cd2_plot.m
3  %
4
5  clear;
6
7  cd2_init;
8
9  u = [1 1 1 -3 -3 2 5 0];
10 t = 0:(size(u,2)-1); % start at zero
11
12 figure;
13 subplot(2,1,1);
14 stairs(t,u);
15 grid on;
16 axis auto;
17 title('Input Signal');
18 ylabel('u');
19
20 subplot(2,1,2);
21 lsim(sys, u);
22 grid on;
23 axis auto;
24
25 print('cd2_plot.eps', '-deps');
```

Listing 4: Matlab code to plot the simplified steady state to delta system.

B Delta to Steady State Transform Derivation

To convert a delta input to a steady state output it should sum the history of values. Figure 8 shows the system.

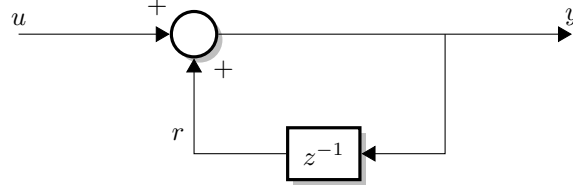


Figure 8: System to convert delta input to steady state output.

This system can be simplified in to a single transfer function as given by Equation 6 and shown in Figure 9.

$$\begin{aligned} y &= u + r \\ r &= y \cdot z^{-1} \\ y &= u + yz^{-1} \\ y &= y(1 - z^{-1}) \end{aligned}$$

$$\boxed{\frac{y}{u} = \frac{1}{1 - z^{-1}}}$$

(6)

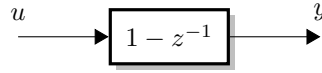


Figure 9: Simplified system to convert a delta input to a steady state output.

Figure 10 shows the response of this system given an arbitrary input ⁴. It can be seen that the output is held in a steady state according to the delta inputs as expected.

⁴This input is actually the output of the steady state input to delta output given in Appendix A. They are equal and opposite as expected.

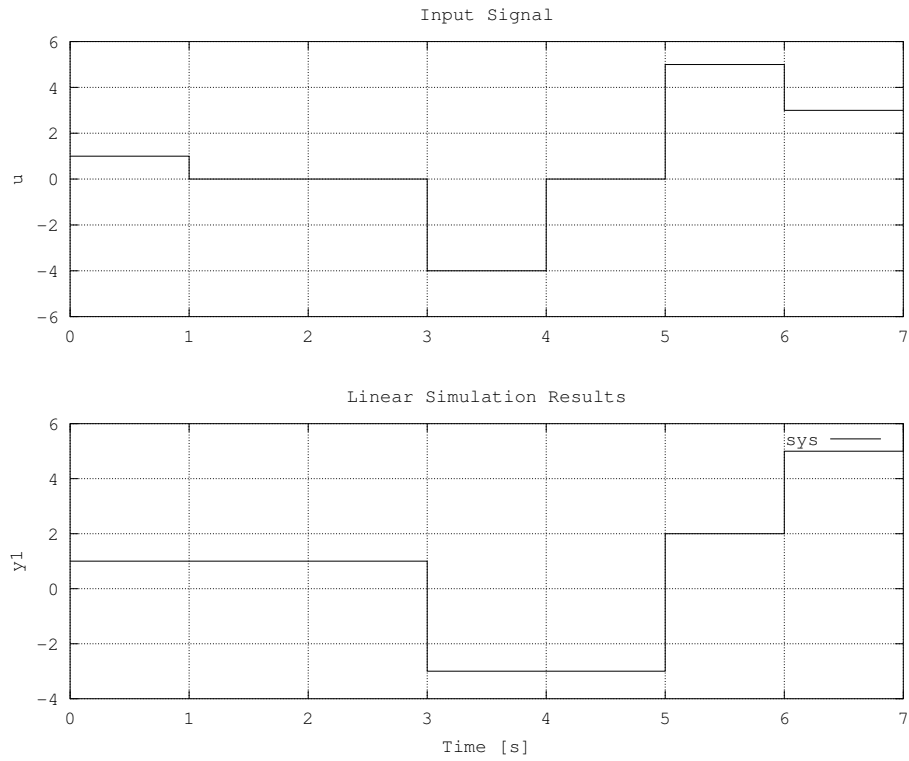


Figure 10: Response of delta input to steady state output system when given an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 5 and 6.

B.1 Matlab Source

The following code has been tested using Octave,⁵ an open source Matlab clone.

```
1 %  
2 % dc1_init.m  
3 %  
4 % Convert a delta input to a steady state output.  
5 %  
6 % Simplified System.  
7 %  
8  
9 T = 1; % time step  
10  
11 sys = tf([1 0], [1 -1], T, 'inname', 'u1', 'outname', 'y1');
```

Listing 5: Matlab code to initialize the simplified steady state to delta system.

```
1 %  
2 % dc2_plot.m  
3 %  
4  
5 clear;  
6  
7 dc2_init;  
8  
9 u = [1 0 0 -4 0 5 3 3];  
10 t = 0:(size(u,2)-1); % start at zero  
11  
12 figure;  
13 subplot(2,1,1);  
14 stairs(t,u);  
15 grid on;  
16 %axis auto;  
17 axis([0 7 -6 6]);  
18 title('Input Signal');  
19 ylabel('u');  
20  
21 subplot(2,1,2);  
22 lsim(sys, u);  
23 grid on;  
24 axis auto;  
25 axis([0 7 -4 6]);  
26  
27 print('dc2_plot.eps', '-deps');
```

Listing 6: Matlab code to plot the simplified steady state to delta system.

⁵Octave community, see n. 3.

C Linear Engine Model Plot Matlab Script

```
1 clear all;
2 clf;
3
4 linear_engine_model_init;
5
6 sim('linear_engine_model');
7
8 figure;
9
10
11
12 subplot(3,1,1);
13 plot(dout.Time, dout.Data(:,1));
14 grid on;
15 axis tight;
16 title('Idle Duty');
17 ylabel('idle duty (%)');
18 xlabel('time (sec)');
19
20 subplot(3,1,2);
21 plot(dout.Time, dout.Data(:,3));
22 grid on;
23 axis tight;
24 title('Torque');
25 ylabel('torque');
26 xlabel('time (sec)');
27
28 subplot(3,1,3);
29 plot(dout.Time, dout.Data(:,2));
30 grid on;
31 axis tight;
32 title('RPM');
33 ylabel('RPM');
34 xlabel('time (sec)');
```

D Linear Engine Model Initialization Matlab Script

```
1 % initialization for the Linear Engine Model
2 % 'linear_engine_model.mdl'
3
4 idle_rpm = 700; % (RPM)
5 num_cyl = 8;
6 % frequency of ignition impulses
7 % In two crankshaft rotations all cylinders will fire
8 % for a 4-stroke engine.
9 idle_rps = 700/60; % minute -> second
10 ign_freq = num_cyl*idle_rps*0.5; % (Hz)
11
12 % engine model time step
13 Tm = 1/ign_freq; % (sec)
14
15 % control time step
16 Tc = 0.5; % (sec)
```