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

Jeremiah Mahler
jmahler@mail.csuchico.edu

CSU Chico
November 29, 2013

DRAFT

1 Engine Model

The engine model used here is based work by Butts and Sivashankar[1] 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.

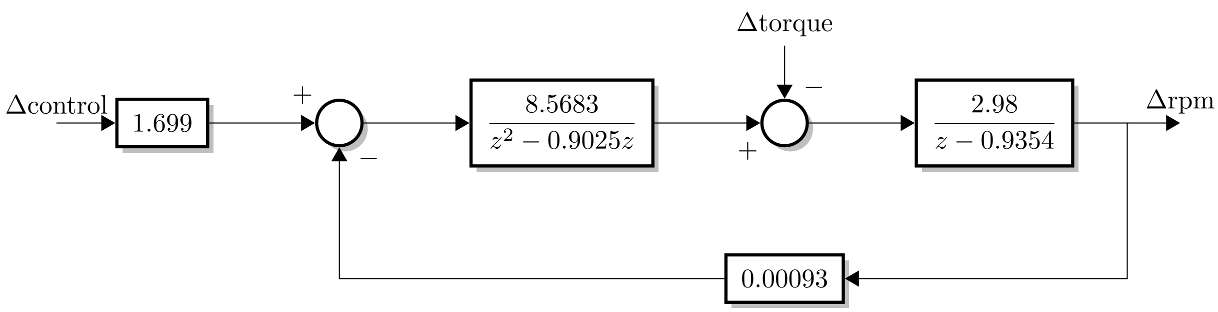

Figure 1: Linear engine model of a modern 4.6L V-8.

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.

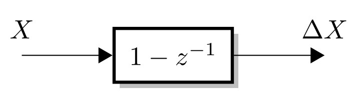

Figure 2: The Z transform used to accumulate the input and convert a steady state input to delta output. Its derivation is given in Appendix A.

[1] K. 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.

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

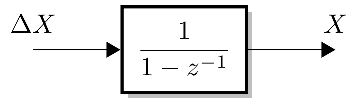

Figure 3: The Z transform used to convert delta input to a steady state output. Its derivation is given in Appendix B.

Typical control systems have an associated time step. And the choice of this time step is crucial in determining performance with regard to the Nyquist frequency. However this model does not suffer from this issue because it is inherently discrete. A single ignition event of the engine corresponds to a single step of the model.

In order to build a controller the model needs to be simplified but the presence of two inputs complicates matters. To resolve this issue the torque can be set zero. With this change it can be simplified by first recognizing that it matches the form shown in Figure 4 which has the transfer function in Equation 1.

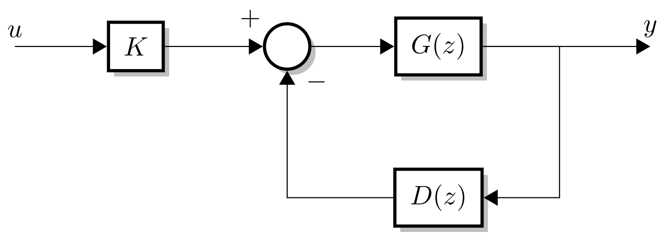

Figure 4: Direct Design system with K as a scaling input, G is the plant and D is the controller.

$$\frac{y}{u} = K \frac{B\alpha}{A\alpha + B\beta} \tag{1}$$

First taking the parts from the engine model (Figure 1).

$$\begin{aligned}
\frac{B}{A} &= \frac{8.5683}{(z^2 - 0.9025z)} \frac{2.98}{(z - 0.9354)} \\
K &= 1.699 \\
\frac{\beta}{\alpha} &= 0.00093
\end{aligned}$$

Then substituting them in to Equation 1 results in the simplified from of Equation 2.

$$\begin{aligned}
\frac{y}{u} &= \frac{(1.699)(8.5683)(2.98)(1)}{(z^2 - 0.9025z)(z - 0.9354)(1) + (8.5683)(2.98)(0.00093)} \\
&\boxed{\frac{y}{u} = \frac{43.38}{z^3 - 1.838z^2 + 0.8442z + 0.02375}}
\end{aligned} \tag{2}$$

It is still necessary to address the delta inputs and outputs. It can be seen in Figure 5 there is a beneficial side effect that makes this trivial. The transform that converts from steady state to a delta cancels with the transform that converts from a delta to steady state [3]. Therefore the final transform is still Equation 2. An example of the output response with no control is shown in Figure 6.

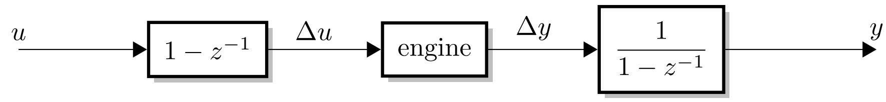

Figure 5: Engine model with transforms for converting from steady state to delta and vice versa.

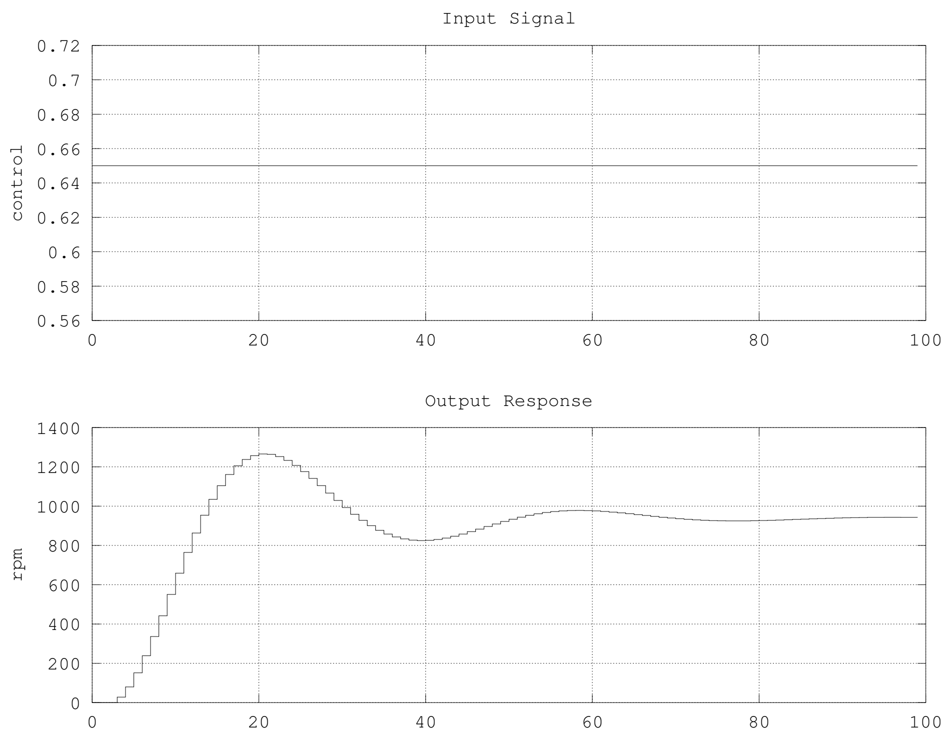

Figure 6: Output response of engine model with no control. The control can range from zero to one. Here the control is constant at 0.65 which resulted in a stable rpm near 900. The Matlab script is provided in Appendix C.

[3]If the torque input was non zero the transforms would not cancel.

2 Regulator Control System

What is needed is a regulator system where the input is applied as torque. The general system is shown in Figure 7.

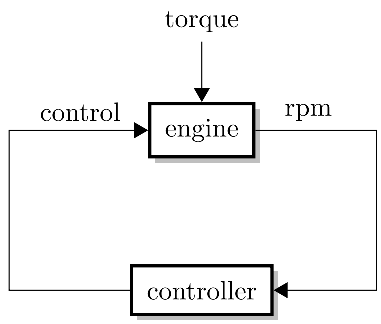

Figure 7: General system for regulating engine rpm.

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 8. 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).

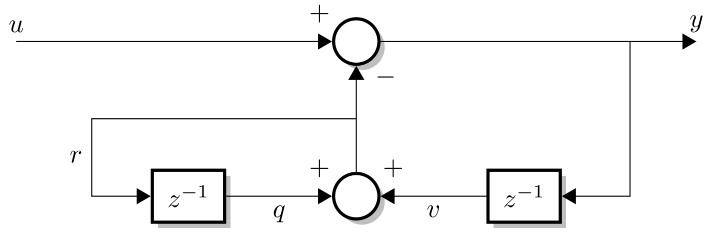

Figure 8: System to accumulate values to convert a steady state input to a delta output.

Figure 9 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.

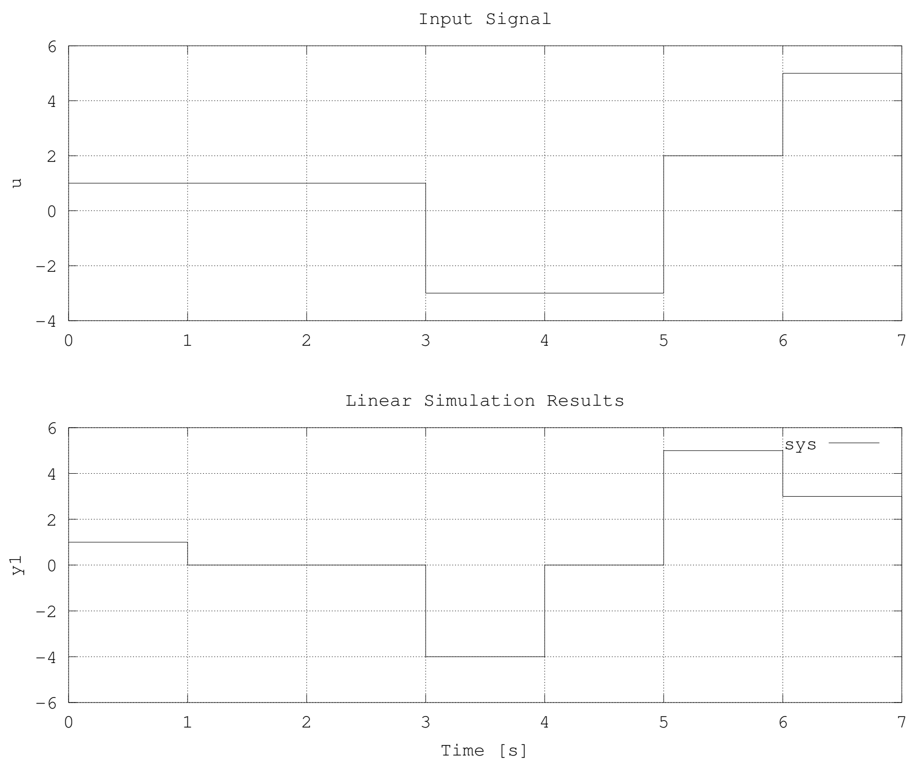

Figure 9: Response of full steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 1 and 2.

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

$$
\begin{aligned}
r &= q + v && (3)\\
v &= y \cdot z^{-1} && (4)\\
q &= r \cdot z^{-1} && (5)\\
y &= u - r && (6)
\end{aligned}
$$

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

$$
\begin{aligned}
r &= rz^{-1} + yz^{-1} && (3,4,5)\\
r(1 - z^{-1}) &= yz^{-1} &&\\
r &= u - y && \text{(6}\\
(u - y)(1 - z^{-1}) &= yz^{-1} &&\\
u - y - uz^{-1} + yz^{-1} &= yz^{-1} &&\\
u - y - uz^{-1} &= 0 &&\\
y &= u(1 - z^{-1}) &&
\end{aligned}
$$

$$\boxed{\frac{y}{u} = 1 - z^{-1}} \qquad (7)$$

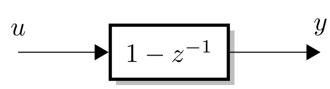

Figure 10: Simplified system to convert a steady state input in to a delta output.

It can be seen in Figure 11 that the simplified system behaves identically to the previous system (Figure 9).

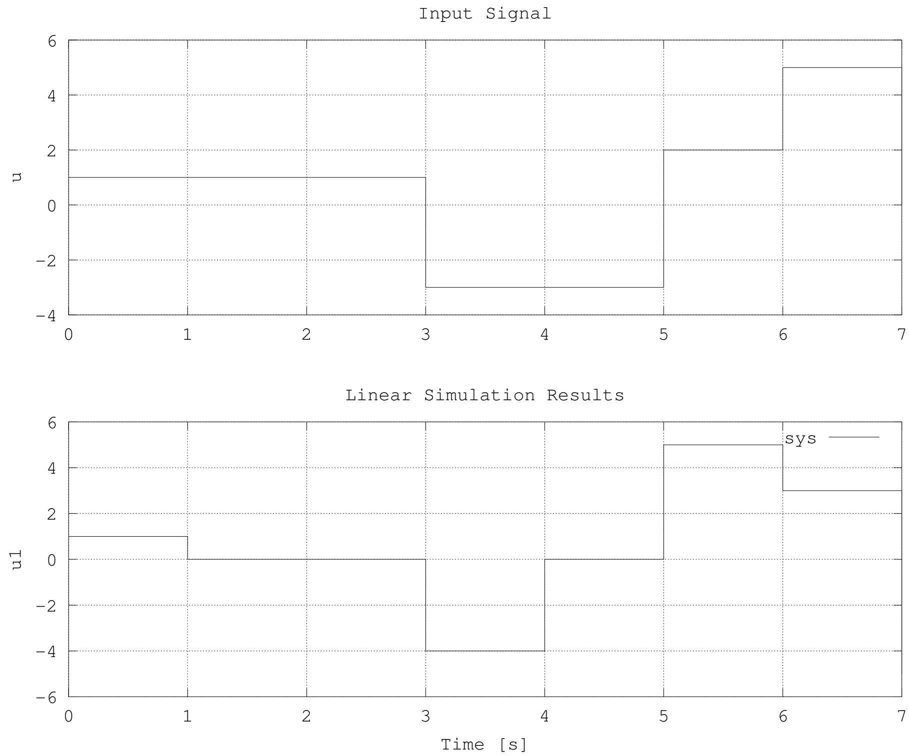

Figure 11: Response of simplified steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). Response is identical to the full system in Figure 9 as expected. The Matlab source code is given in Listing 3 and 4.

A.1 Matlab Source

The following code has been tested using Octave,[4] an open source Matlab clone.

```
%
% cd1_init.m
%
% Convert a steady state input to a delta output.
%
% Full System.
%

T = 1;  % time step

D1 = tf([1], [1 0], T, 'inname', 'y1', 'outname', 'v1');
D2 = tf([1], [1 0], T, 'inname', 'r1', 'outname', 'q1');
sum1 = sumblk('y1 = u1 - r1');
sum2 = sumblk('r1 = v1 + q1');
sys = connect(D1, D2, sum1, sum2, 'u1', 'y1');
```

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

```
%
% cd1_plot.m
%

clear;

cd1_init;

u = [1 1 1 -3 -3 2 5 0];
t = 0:(size(u,2)-1);  % start at zero

figure;
subplot(2,1,1);
stairs(t,u);
grid on;
axis auto;
title('Input Signal');
ylabel('u');

subplot(2,1,2);
lsim(sys, u);
grid on;
axis auto;

print('cd1_plot.eps', '-deps');
```

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

[4] Octave community. *GNU/Octave*. 2012. URL: `www.gnu.org/software/octave/`.

```
%
% cd2_init.m
%
% Convert a steady state input to a delta output.
%
% Simplified System.
%

T = 1;  % time step

sys = tf([1 -1], [1 0], T, 'inname', 'y1', 'outname', 'u1');
```

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

```
%
% cd2_plot.m
%

clear;

cd2_init;

u = [1 1 1 -3 -3 2 5 0];
t = 0:(size(u,2)-1);  % start at zero

figure;
subplot(2,1,1);
stairs(t,u);
grid on;
axis auto;
title('Input Signal');
ylabel('u');

subplot(2,1,2);
lsim(sys, u);
grid on;
axis auto;

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 12 shows the system.

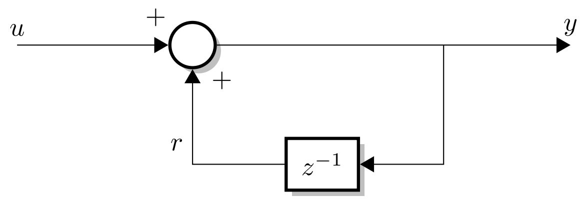

Figure 12: System to convert delta input to steady state output.

This system can be simplified in to a single transfer function as given by Equation 8 and shown in Figure 13.

$$
\begin{aligned}
y &= u + r \\
r &= y \cdot z^{-1} \\
y &= u + yz^{-1} \\
u &= y(1 - z^{-1}) \\
&\boxed{\frac{y}{u} = \frac{1}{1 - z^{-1}}}
\end{aligned}
\tag{8}
$$

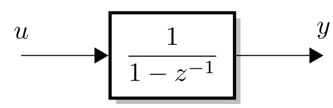

Figure 13: Simplified system to convert a delta input to a steady state output.

Figure 14 shows the response of this system given an arbitrary input [5]. It can be seen that the output is held in a steady state according to the delta inputs as expected.

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

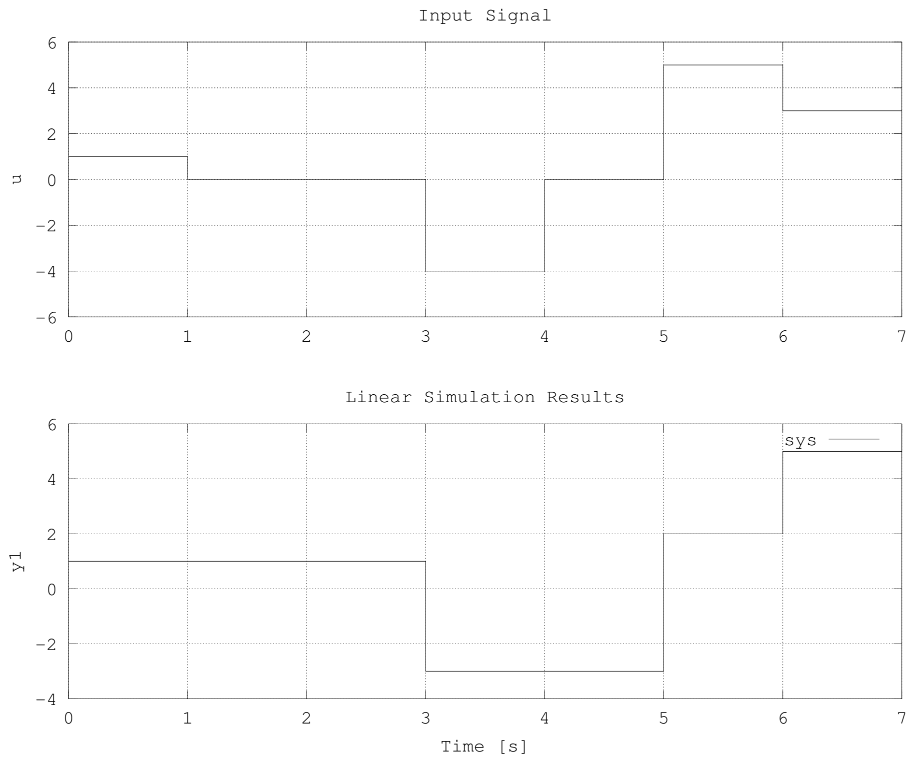

Figure 14: Response of delta input to steady state output system when given an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 5 and 6.

B.1 Matlab Source

The following code has been tested using Octave,[6] an open source Matlab clone.

```
%
% dc1_init.m
%
% Convert a delta input to a steady state output.
%
% Simplified System.
%

T = 1;  % time step

sys = tf([1 0], [1 -1], T, 'inname', 'u1', 'outname', 'y1');
```

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

```
%
% dc2_plot.m
%

clear;

dc2_init;

u = [1 0 0 -4 0 5 3 3];
t = 0:(size(u,2)-1);  % start at zero

figure;
subplot(2,1,1);
stairs(t,u);
grid on;
%axis auto;
axis([0 7 -6 6]);
title('Input Signal');
ylabel('u');

subplot(2,1,2);
lsim(sys, u);
grid on;
axis auto;
axis([0 7 -4 6]);

print('dc2_plot.eps', '-deps');
```

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

[6] Octave community, see n. 4.

C Engine Model Matlab Source

```
%
% engine_model.m
%
% Linear engine model with zero torque.
%

T = 1;  % time step

K = tf([1.699], [1], T);
D1= tf([8.5683], [1 -0.9025 0], T);
D2= tf([2.98], [1 -0.9354], T);
G = D1*D2;
D = tf([0.00093], [1], T);

H = K*G/(1 + D*G);

eng_sys = minreal(H);  % cancel common roots
```

Listing 7: Matlab source to calculate the engine model system with zero torque.

```
%
% engine_model_plot.m
%

clear;

engine_model;

u = [0.65*ones(1,100)];
[y, t, x] = lsim(eng_sys, u);

figure;
subplot(2,1,1);

stairs(t, u);
grid on;
axis auto;
title('Input Signal');
ylabel('control');

subplot(2,1,2);
stairs(t, y);
grid on;
axis auto;
title('Output Response');
ylabel('rpm');

print('engine_model_plot.eps', '-deps');
```

Listing 8: Matlab source to plot the response of engine model with no control.