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 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.

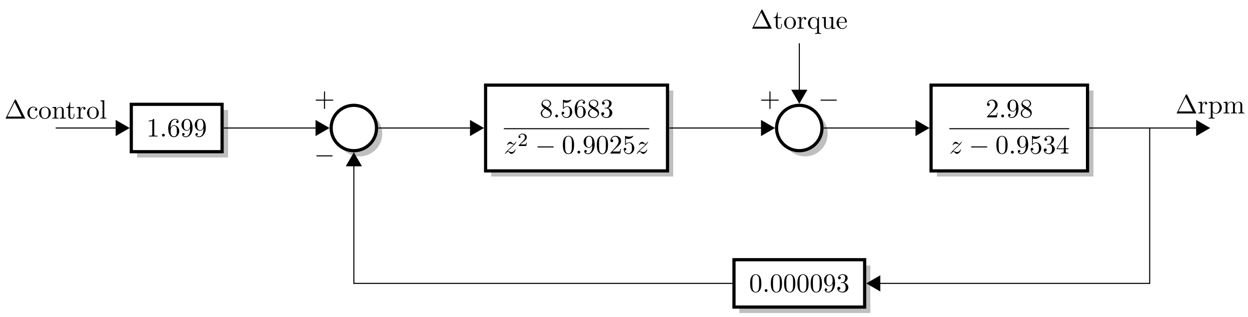

Figure 1: Linear engine model.

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).

This model has one complication that makes it difficult to use compared to typical control systems. All the inputs and outputs are specified as deltas (Δ). The torque input, for example, cannot be given as a constant (T) because this actually means ΔT

To overcome this complication a simple feedback controller can be used as show in Figure 2. And the open loop controller, with a fixed input duty cycle, is shown in Figure 3. Notice that in this example the torque has transient characteristics and varies over time. The output of this system is show in Figure 4.

[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.

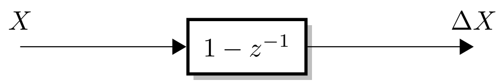

Figure 2: Z transform used to accumulate the input and convert a steady state input to delta output. For its derivation, see Appendix A.

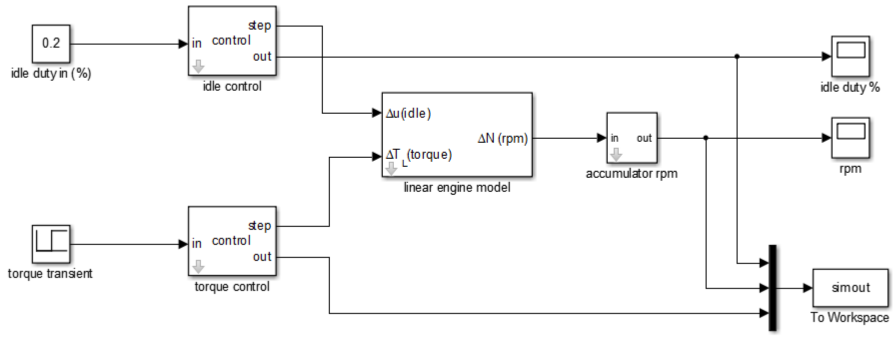

Figure 3: Linear engine model with an open loop controller. Inputs go through the simple controller to provide steps to the engine model. The output rpm, since it is in steps (ΔN), is accumulated.

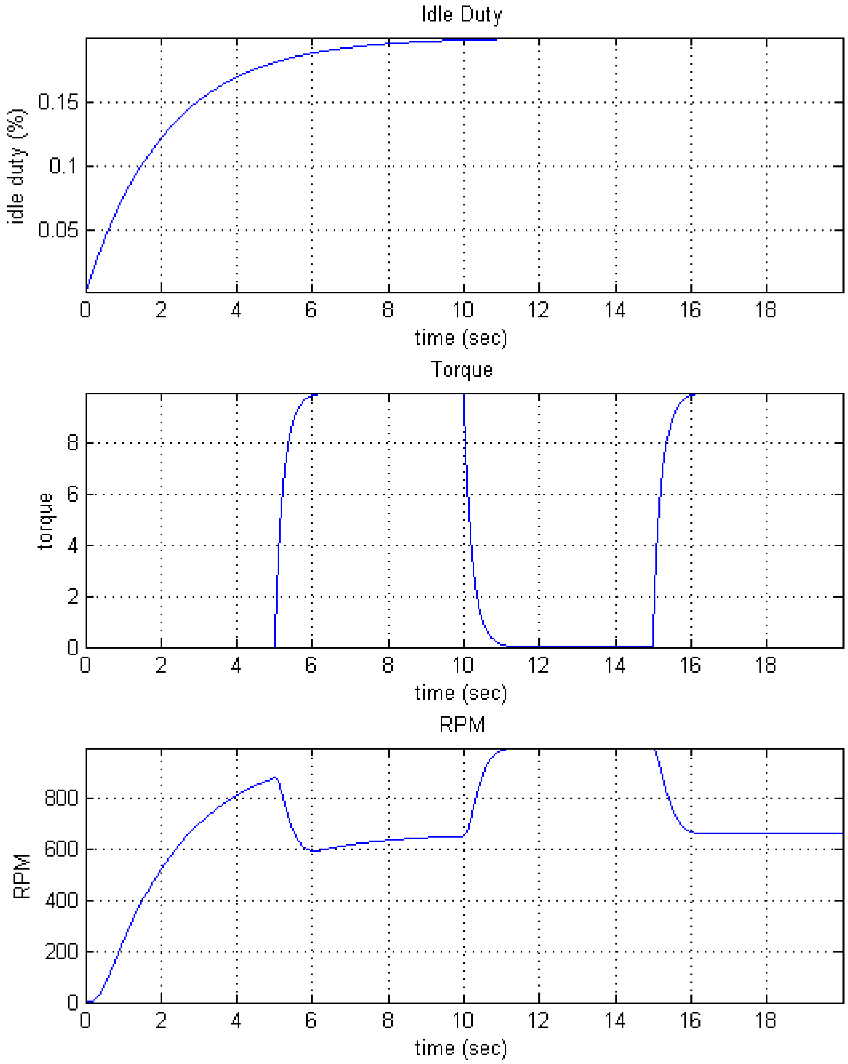

Figure 4: Output of linear engine controller with a transient torque input and a fixed input duty cycle. Matlab source given in Appendix B

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 Z Transform Derivation

To accumulate a steady state input to produce a delta output a system can be constructed as shown in Figure 5. 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).

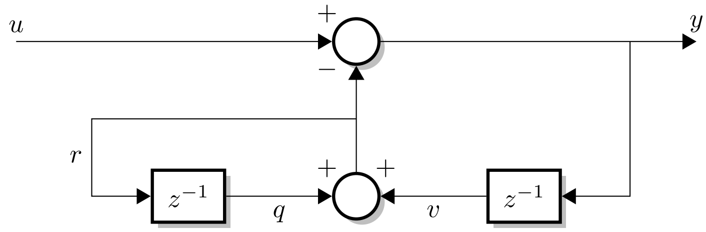

Figure 5: System to accumulate values to convert a steady state input to a delta output.

Figure 6 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.

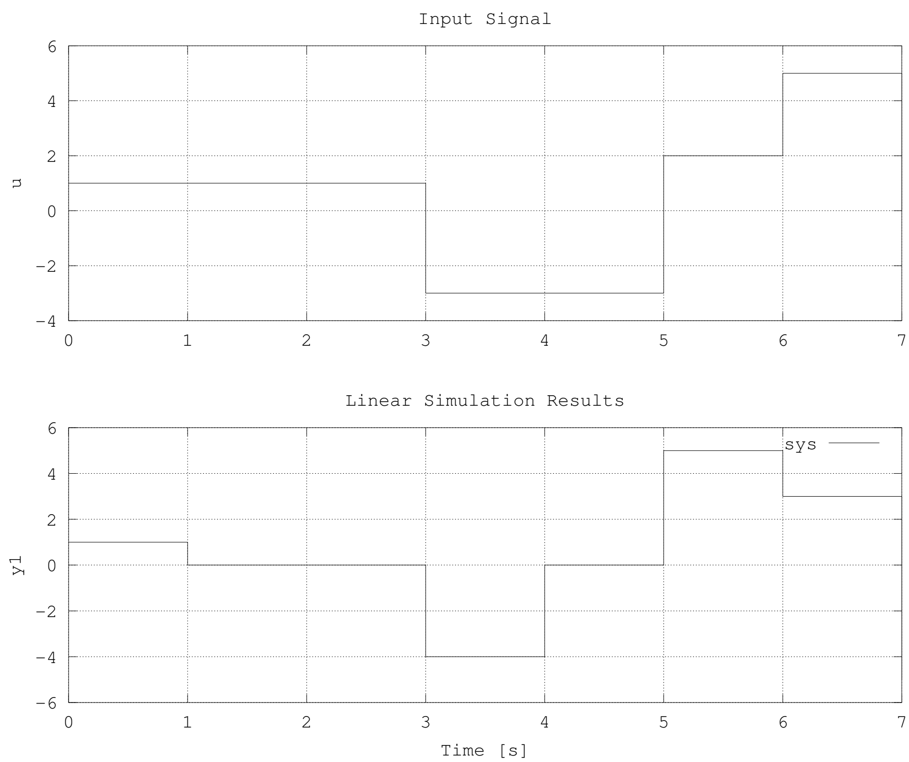

Figure 6: Response of full steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 1 and 2.

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

$$r = q + v \tag{1}$$
$$v = y \cdot z^{-1} \tag{2}$$
$$q = r \cdot z^{-1} \tag{3}$$
$$y = u - r \tag{4}$$

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

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

$$\boxed{\frac{y}{u} = 1 - z^{-1}} \tag{5}$$

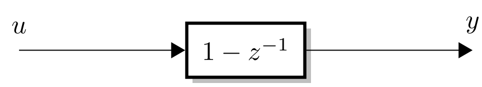

Figure 7: Simplified system to convert a steady state input in to a delta output.

It can be seen in Figure 8 that the simplified system behaves identically to the previous system (Figure 6).

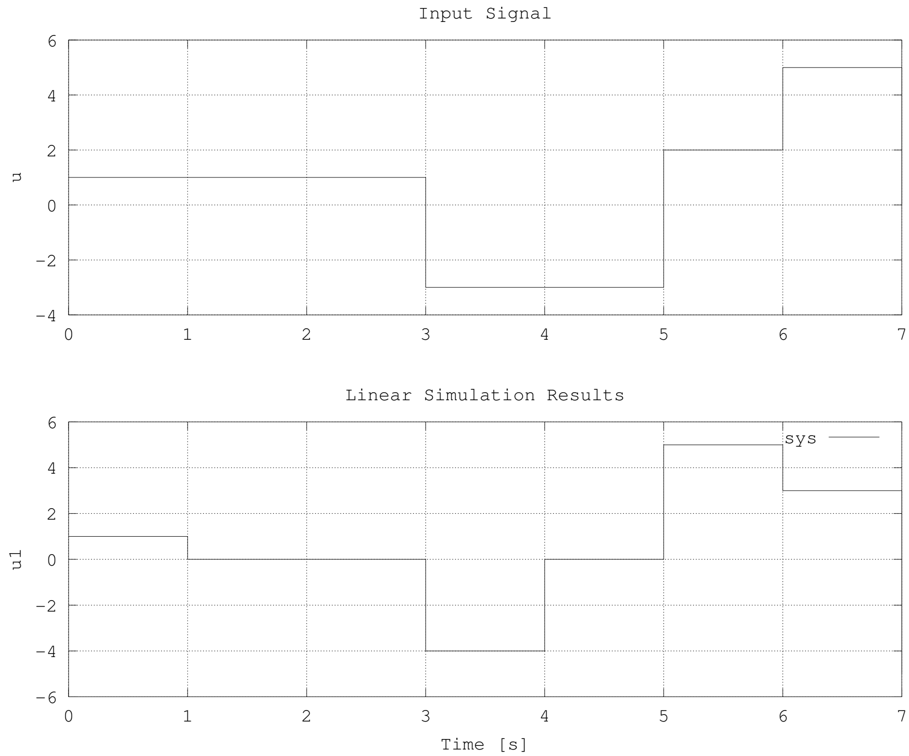

Figure 8: Response of simplified steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). Response is identical to the full system in Figure 6 as expected. The Matlab source code is given in Listing 3 and 4.

A.1 Matlab Source

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

```
%
% 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.

[3] 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 Linear Engine Model Plot Matlab Script

```matlab
clear all;
clf;

linear_engine_model_init;

sim('linear_engine_model');

figure;



subplot(3,1,1);
plot(dout.Time, dout.Data(:,1));
grid on;
axis tight;
title('Idle Duty');
ylabel('idle duty (%)');
xlabel('time (sec)');

subplot(3,1,2);
plot(dout.Time, dout.Data(:,3));
grid on;
axis tight;
title('Torque');
ylabel('torque');
xlabel('time (sec)');

subplot(3,1,3);
plot(dout.Time, dout.Data(:,2));
grid on;
axis tight;
title('RPM');
ylabel('RPM');
xlabel('time (sec)');
```

C Linear Engine Model Initialization Matlab Script

```matlab
% initialization for the Linear Engine Model
% 'linear_engine_model.mdl'

idle_rpm = 700;  % (RPM)
num_cyl = 8;
% frequency of ignition impulses
% In two crankshaft rotations all cylinders will fire
% for a 4-stroke engine.
idle_rps = 700/60;  % minute -> second
ign_freq = num_cyl*idle_rps*0.5;  % (Hz)

% engine model time step
Tm = 1/ign_freq;  % (sec)

% control time step
Tc = 0.5;  % (sec)
```