Control System Design Applied to Idle Stabilization of a Spark Ignition Engine

Jeremiah Mahler
jmahler@mail.csuchico.edu

CSU Chico
December 4, 2013

DRAFT

Abstract

The task of maintaining a stable idle for an internal combustion engine with spark ignition is non-trivial. Any time an accessory is turned on/off the torque applied to the engine changes. And changes in torque will change the engine rpm if the control inputs are constant. This paper shows how control system methods can be applied to the problem of idle stabilization. Because the engine model is inherently discrete all of the methods used are also discrete. Methods include: pole placement, direct design, and various state space designs.

1 Engine Model

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

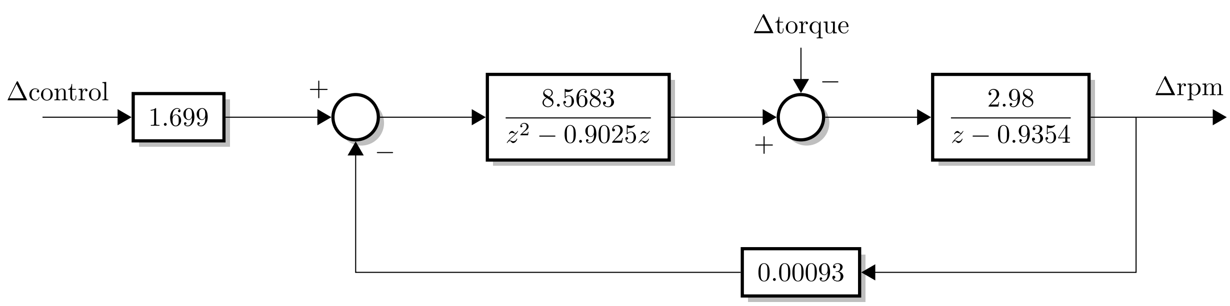

Figure 1: Linear engine model of a modern 4.6L V-8.

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.

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

[3] The derivation of the steady state to delta conversion is given in Appendix A

[4] The derivation of the delta to steady state conversion is given in Appendix B.

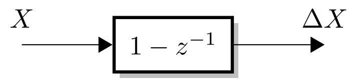

Figure 2: The Z transform used to accumulate the input and convert a steady state input to delta output.

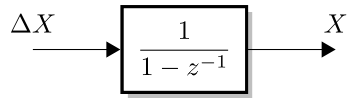

Figure 3: The Z transform used to convert delta input to a steady state output.

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 construct a controller the model (Figure 1) needs to be simplified in to a transfer function. However the prescence of two inputs, control and torque, complicates matters. To resolve this issue the torque can be set zero. Then it can be simplified by recognizing that it matches the well known form shown in Figure 4 which has the transfer function in Equation 1.

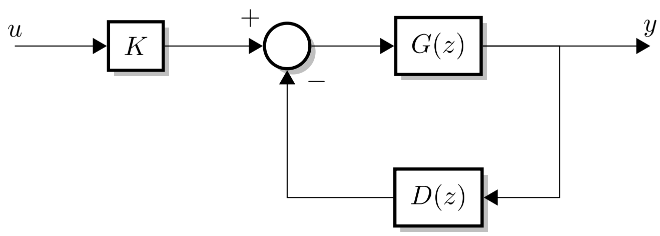

Figure 4: Direct Design system with K as a scaling input, G is the plant and D is the controller.

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

To simply first gather 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 substitute them in to Equation 1 and simplify. The result is Equation 2.

$$\frac{y}{u} = \frac{(1.699)(8.5683)(2.98)(1)}{(z^2 - 0.9025z)(z - 0.9354)(1) + (8.5683)(2.98)(0.00093)}$$

$$\boxed{\frac{y}{u} = \frac{43.38}{z^3 - 1.838z^2 + 0.8442z + 0.02375}} \tag{2}$$

It is still necessary to address the delta inputs and outputs. This can be resolved by placing the conversion transforms (from Figure 2 and 3) on either end of the engine model as shown in Figure 5. A beneficial side effect becomes apparent in this form. The transform that converts from steady state to a delta cancels with the transform that converts from a delta to steady state [5]. Therefore the final transform is still Equation 2. An example of the output response with no control is shown in Figure 6. The Matlab source code for this plot and others are in Appendix C.

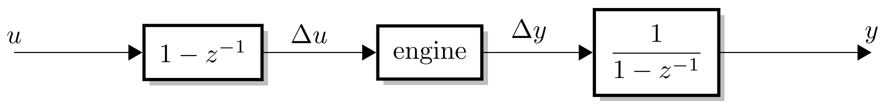

Figure 5: Engine model with transforms for converting from steady state to delta and vice versa.

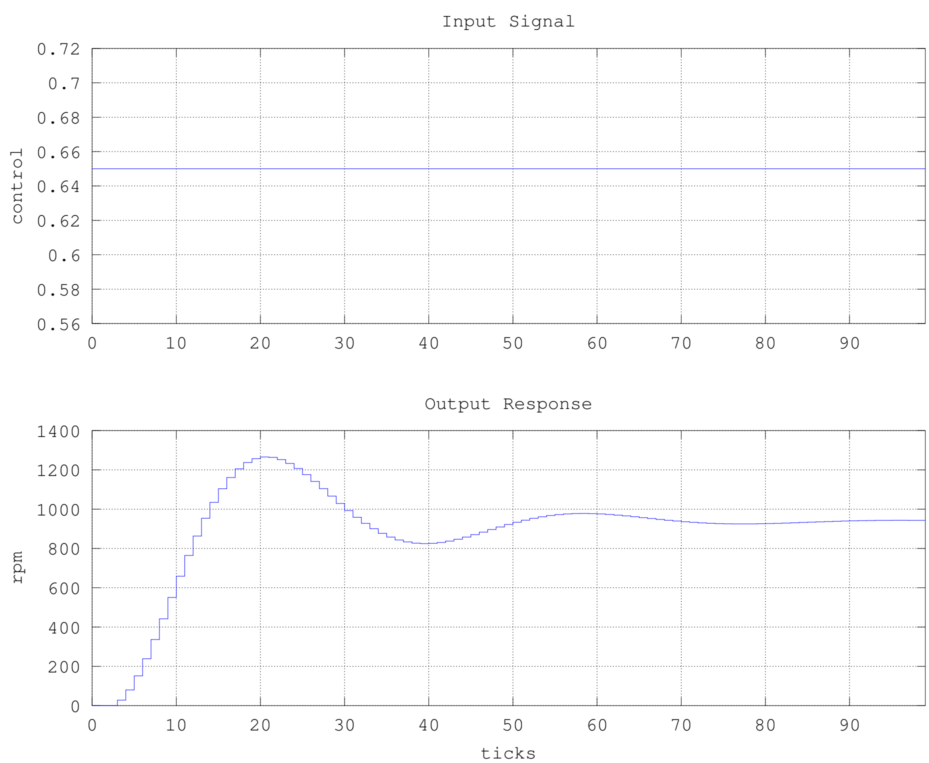

Figure 6: Output response of engine model with no control. The control can range from zero to one. Here the control is constant at 0.65 which resulted in a stable rpm near 900.

It is not apparent from Figure 6 exactly how long this process takes because the x axis represents ignition events instead of time. However the rpm can be converted to time per ignition event by examining the units as shown in Equation 3.

[5] In the more general case, when torque is not zero, the transforms wouldn't cancel.

$$\frac{1}{x\frac{\text{rev}}{\text{min}} \cdot n\frac{\text{ign}}{\text{rev}} \cdot \frac{\text{min}}{60\text{sec}}} = y\frac{\text{sec}}{\text{ign}} \tag{3}$$

$\frac{\text{rev}}{\text{min}}$: revolutions per minute (rpm)

$\frac{\text{ign}}{\text{rev}}$: number ignition points per revolution (4 for 8 cylinder engine)

$\frac{\text{sec}}{\text{ign}}$: seconds per ignition point

Applying Equation 3 results in the response shown in Figure 7. It can be seen that the rise time is over 0.5 second and that the overshoot is close to 40%. The stabilization time is also several seconds. Clearly its performance could be improved with the application of a controller.

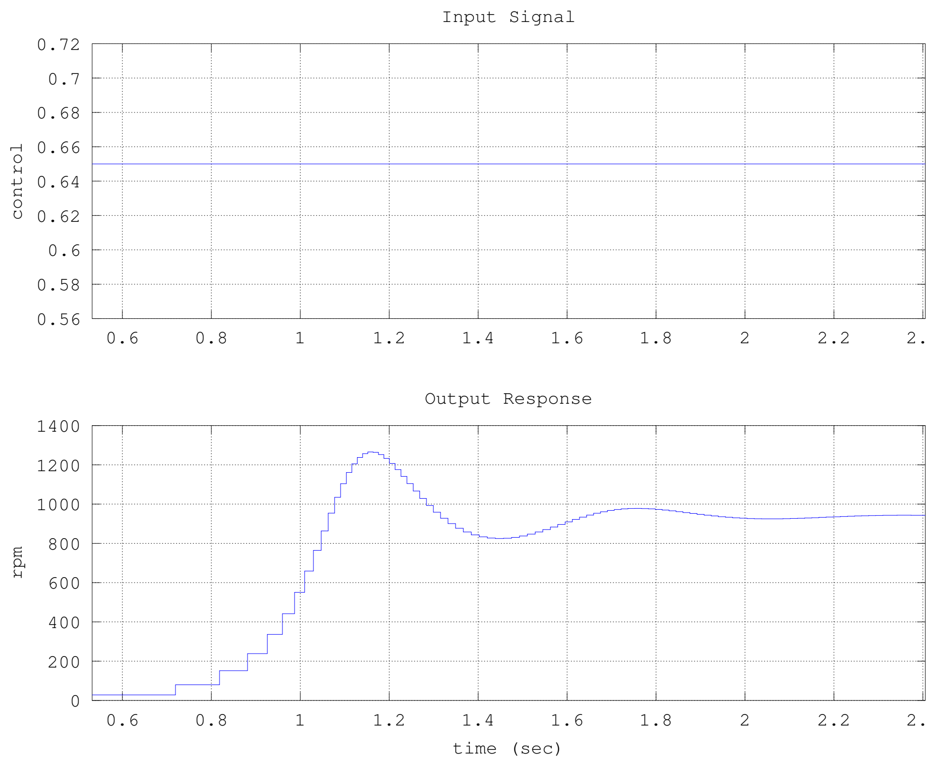

Figure 7: Engine rpm response compared to time with a constant 0.65 input control.

The response from zero rpm is not a realistic measurement of performance because zero rpm is impossibly slow. A typical idle rpm range is from 600 to 1000 rpm. Figure 8 shows the response for a step input with control inputs chosen to reach 600 and 1000 rpm [6]. This is a more realistic representation of typical operation. The rise time is near 0.2 seconds and the overshoot is 15%. And the stabilization time is almost one second. Its behavior can still be improved with a controller.

[6]The control input to stable rpm values are given in Appendix E

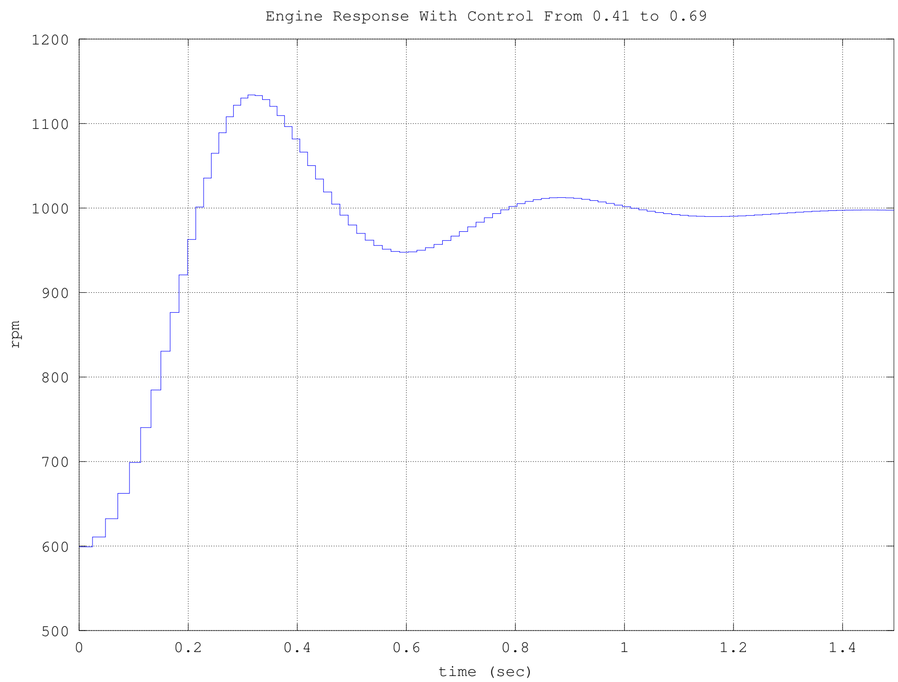

Figure 8: Response for a shifted and scaled step input with zero torque.

2 Direct Design

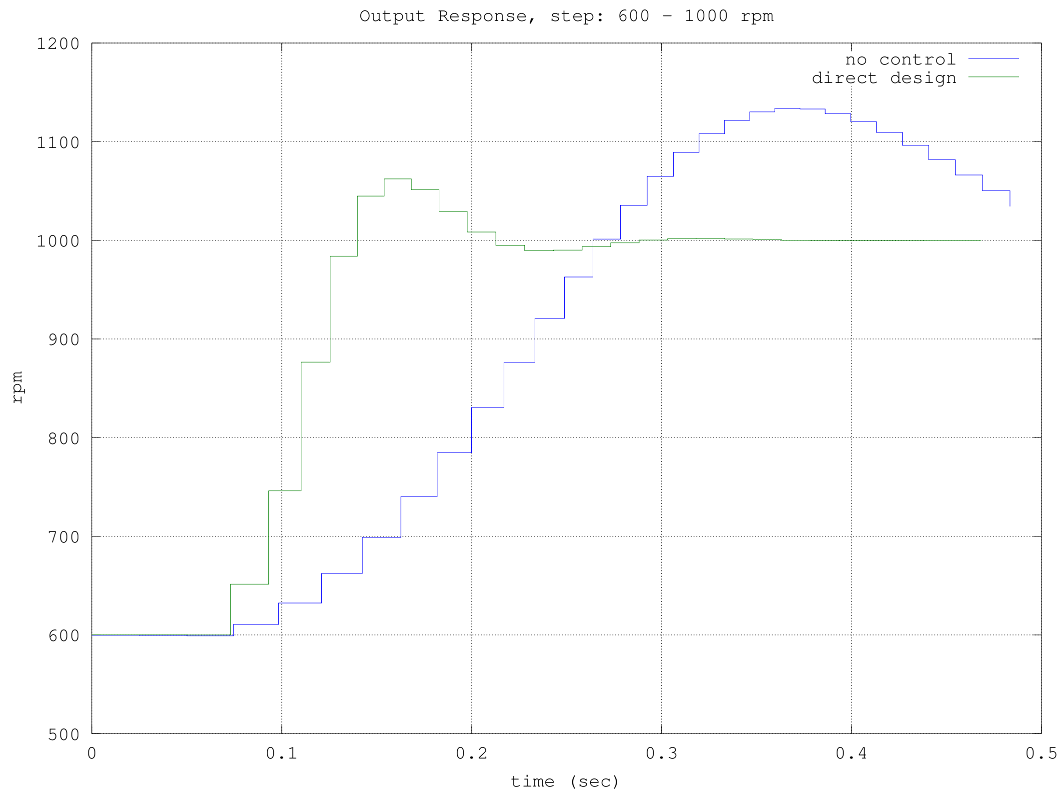

Figure 9: Output response of controller built using Direct Design compared to no control. For no control this is the same plot as in Figure 8 which stabilizes after a second.

3 Regulator Control System

What is needed is a regulator system where the input is applied as torque. The general system is shown in Figure 10.

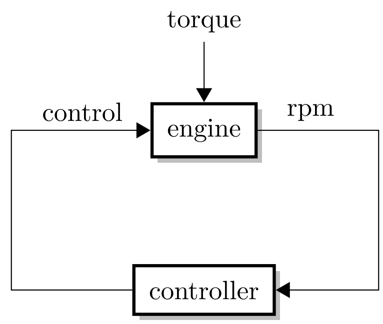

Figure 10: General system for regulating engine rpm.

4 Conclusion

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 11. 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).

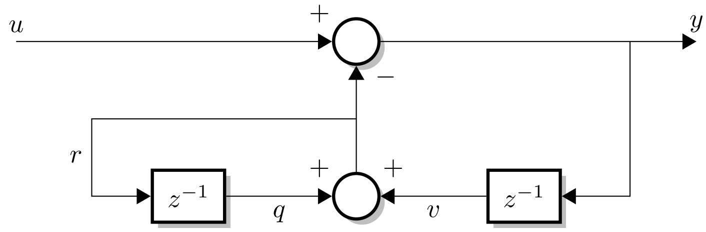

Figure 11: System to accumulate values to convert a steady state input to a delta output.

Figure 12 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.

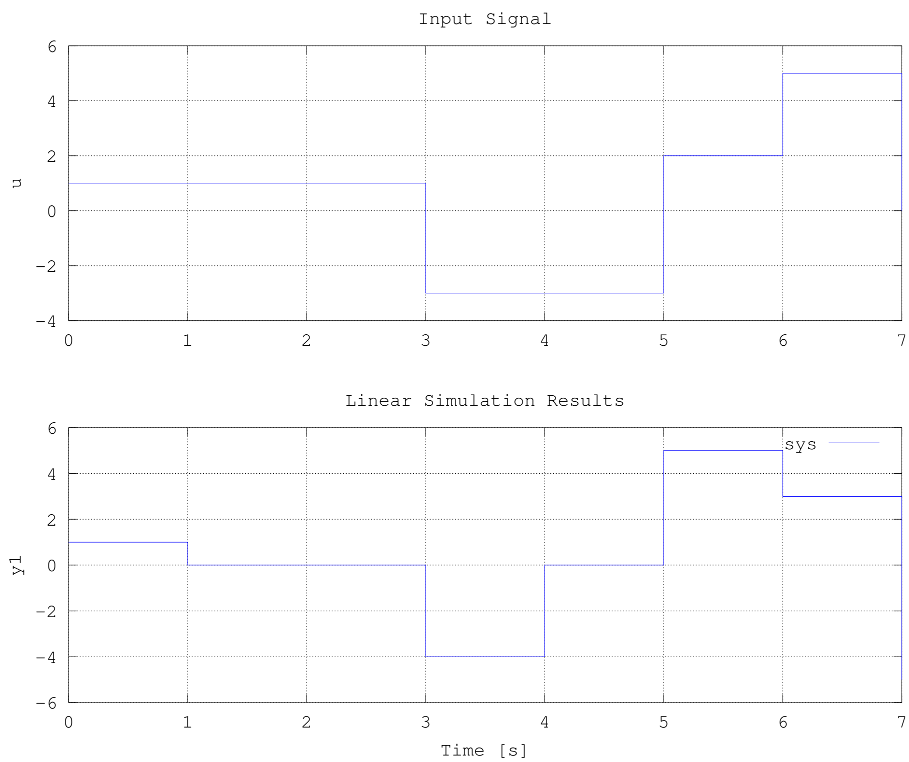

Figure 12: Response of full steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 1.

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

$$r = q + v \tag{4}$$
$$v = y \cdot z^{-1} \tag{5}$$
$$q = r \cdot z^{-1} \tag{6}$$
$$y = u - r \tag{7}$$

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

$$\begin{aligned}
r &= rz^{-1} + yz^{-1} & & (4,5,6) \\
r(1 - z^{-1}) &= yz^{-1} & & \\
r &= u - y & & (7 \\
(u - y)(1 - z^{-1}) &= yz^{-1} & & \\
u - y - uz^{-1} + yz^{-1} &= yz^{-1} & & \\
u - y - uz^{-1} &= 0 & & \\
y &= u(1 - z^{-1}) & &
\end{aligned}$$

$$\boxed{\frac{y}{u} = 1 - z^{-1}} \tag{8}$$

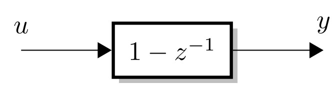

Figure 13: Simplified system to convert a steady state input in to a delta output.

It can be seen in Figure 14 that the simplified system behaves identically to the previous system (Figure 12).

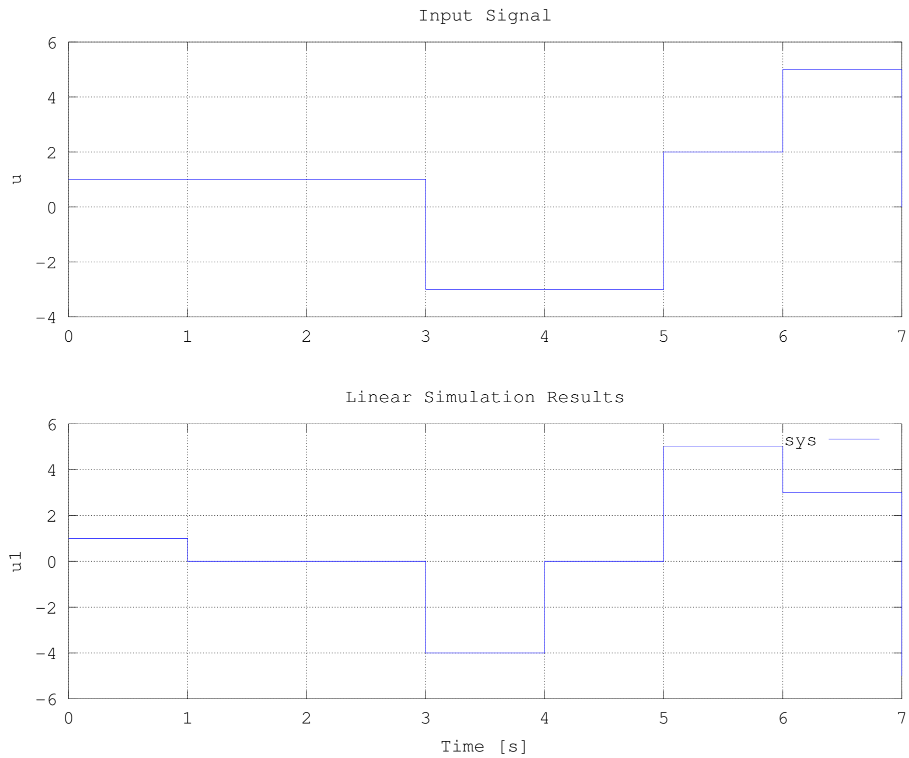

Figure 14: Response of simplified steady state input to delta output system to an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). Response is identical to the full system in Figure 12 as expected. The Matlab source code is given in Listing 2.

A.1 Matlab Source

The following code has been tested using Octave,[7] an open source Matlab clone.

```
%
% cd_plot1.m
%

clear;

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');

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('cd_plot1.eps', '-color', '-deps2');
```

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

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

```matlab
%
% cd2_plot.m
%

clear;

T = 1;  % time step

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

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('cd_plot2.eps', '-color', '-deps2');
```

Listing 2: 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 15 shows the system.

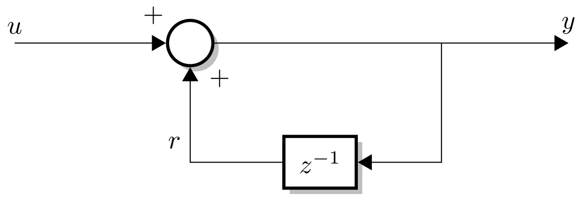

Figure 15: System to convert delta input to steady state output.

This system can be simplified in to a single transfer function as given by Equation 9 and shown in Figure 16.

$$
\begin{aligned}
y &= u + r \\
r &= y \cdot z^{-1} \\
y &= u + yz^{-1} \\
u &= y(1 - z^{-1}) \\
\boxed{\frac{y}{u} = \frac{1}{1 - z^{-1}}}
\end{aligned}
\tag{9}
$$

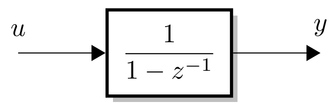

Figure 16: Simplified system to convert a delta input to a steady state output.

Figure 17 shows the response of this system given an arbitrary input [8]. It can be seen that the output is held in a steady state according to the delta inputs as expected.

[8]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.

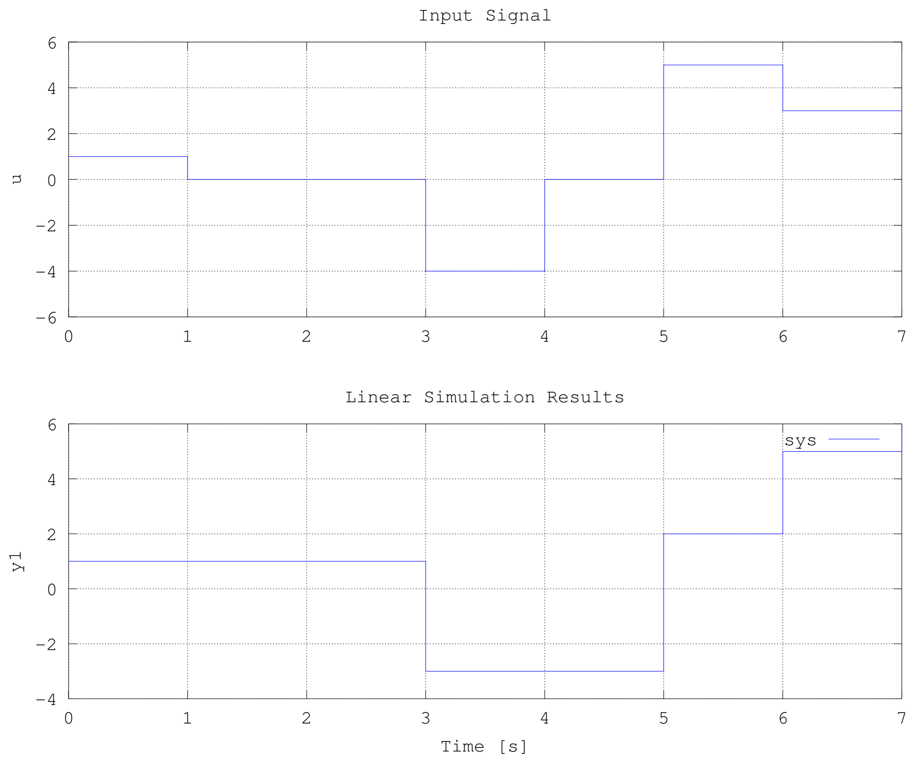

Figure 17: Response of delta input to steady state output system when given an arbitrary input signal. The upper plot is the input signal (u) and the lower plot is the output response (y). The Matlab source code is given in Listing 3.

B.1 Matlab Source

The following code has been tested using Octave,[9] an open source Matlab clone.

```
%
% dc_plot2.m
%

clear;

T = 1;  % time step

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

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('dc_plot2.eps', '-color', '-deps2');
```

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

[9]Octave community, see n. 7.

C Engine Model Matlab Source

```
function [sys] = engine_model(T);
%  ENGINE_MODEL
%
% 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);

sys = minreal(H);  % cancel common roots

endfunction
```

Listing 4: Matlab source to calculate the engine model system with zero torque. This function is called by other scripts that use this model.

```
%
% em_plot.m
%
% Plot of engine model with zero torque and no
% control.  x axis is in ticks (not time).
%

clear;

T = 1;  % time step
Gz = engine_model(T);

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

figure;
subplot(2,1,1);

stairs(t, u);
grid on;
axis([t(1) t(end)]);
title('Input Signal');
ylabel('control');

subplot(2,1,2);
stairs(t, y);
grid on;
axis([t(1) t(end)]);
title('Output Response');
ylabel('rpm');
xlabel('ticks');

print('em_plot1.eps', '-color', '-deps2');
```

Listing 5: Matlab source to plot the response of engine model with no control. The x axis is in ticks.

```
%
% em_plot2.m
%
% Engine model with no control and x axis in time.
%

clear;

ncyl = 8;

T = 1;  % time step
Gz = engine_model(T);

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

% An rpm of zero makes the rpmtime huge and
% skews the results.
% Remove these initial zeros.
i = find(rpm~=0, 1, 'first');
rpm = rpm(i:end);
u = u(i:end);
t = t(i:end);

% Convert each rpm point to time instant.
% The cummulative sum is then the time it takes
% to get to that particular point.
rt = rpmtime(rpm, 8);

figure;
subplot(2,1,1);
stairs(rt, u);
grid on;
axis([rt(1), rt(end)]);
title('Input Signal');
ylabel('control');

subplot(2,1,2);
stairs(rt, rpm);
grid on;
axis([rt(1), rt(end)]);
title('Output Response');
ylabel('rpm');
xlabel('time (sec)');

print('em_plot2.eps', '-color', '-deps2');
```

Listing 6: Matlab source to plot the response of engine model with no control. The x axis is in time.

```
%
% em_plot4.m
%
% Engine model with no control.
% Allowed to stablize before step is applied.
%

T = 1;

Gz = engine_model(T);

% input signal (control)
u_lo = 600/1450.1;
u_hi = 1000/1450.1;
n = 200;  % number of points
u = [u_lo*ones(1,n/2) u_hi*ones(1,n/2)];

% output response
[y, t, x,i] = lsim1(Gz, u);

rt = rpmtime(y, 8);

% remove data before the step
rt = rt((n/2):end);
y = y((n/2):end);

% reset start time to zero
rt = zerotime(rt);

figure(1);

stairs(rt, y);
grid on;
axis([rt(1), rt(end)]);
title('Engine Response With Control From 0.41 to 0.69');
xlabel('time (sec)');
ylabel('rpm');

print('em_plot4.eps', '-color', '-deps2');
```

Listing 7: Matlab source to plot the response of engine model versus time for a typical 600 to 1000 rpm range.

D Direct Design Matlab Source

```
%
% dd_plot1.m
%
% Plot of solution using direct design compared to no control.
%

clear;

T = 1;
ncyl = 8;

% input signal
u_lo = 600;              % rpm before step
u_hi = 1000;    % rpm after step
% different systems take longer to stabilize (time lo)
n1_lo = 30;              % nth tick to step from lo to hi
n2_lo = 100;
n_hi = 30;               % ticks after step
u1 = [u_lo*ones(1,n1_lo) u_hi*ones(1,n_hi)];
% convert to control values (without control)
u2 = [(u_lo/1450.1)*ones(1,n2_lo) (u_hi/1450.1)*ones(1,n_hi)];

% Build a controller, and the resulting system.
Gz1 = engine_model(T);
% These roots are guess, taken from some other example.
roots = [(0.6 + 0.4i) (0.6 - 0.4i) 0 0 0];
order = 3;
limit = 1;
[Hz1, Dz1, K1] = direct_design(Gz1, roots, order, limit);

% System with no control.
Gz2 = engine_model(T);

% output response
[y1, t1, x, i1] = lsim1(Hz1, u1);
[y2, t2, x, i2] = lsim1(Gz2, u2);  % no control
% adjust the step point for any removed from the beginning
i = max([i1 i2]);

% convert rpm to time
rt1 = rpmtime(y1, ncyl);
rt2 = rpmtime(y2, ncyl);

% remove data before step is applied
p1_start = (length(rt1) - n_hi) + 1;
rt1 = rt1(p1_start:end);
y1 = y1(p1_start:end);

p2_start = (length(rt2) - n_hi) + 1;
rt2 = rt2(p2_start:end);
y2 = y2(p2_start:end);

% reset times to start at zero
rt1 = zerotime(rt1);
rt2 = zerotime(rt2);

% plot

clf;
figure(1);

[xs1, ys1] = stairs(rt1, y1);
```

```
[xs2, ys2] = stairs(rt2, y2);
plot(xs2, ys2, xs1, ys1);

grid on;
title(sprintf('Output Response, step: %d - %d rpm', u_lo, u_hi));
ylabel('rpm');
xlabel('time (sec)');
legend('no control', 'direct design');

print('dd_plot2.eps', '-color', '-deps2');
```

Listing 8: Matlab source used to plot the controller built using Direct Design compared to no control.

E Control Response

For a given control input with zero torque the stable rpm values reached are shown in Figure 18. Equation 10 describes its behavior. Listing 9 shows how this was calculated.

$$y = 1450.1x \tag{10}$$

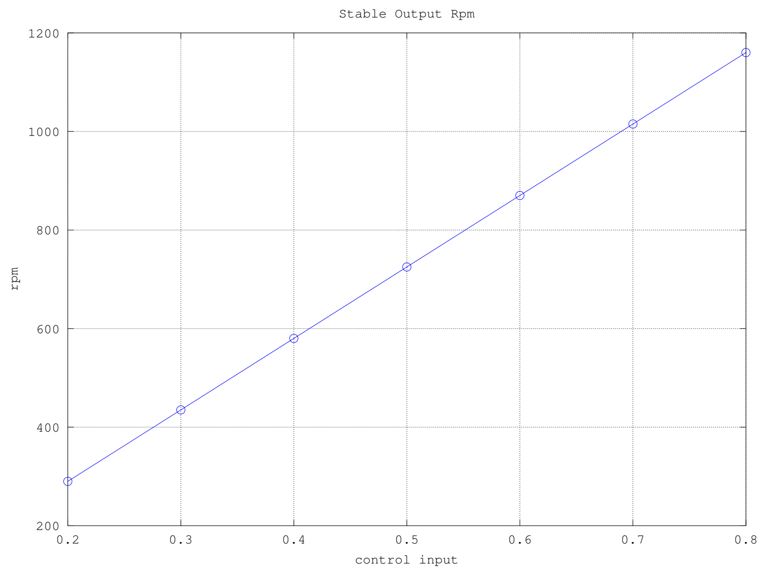

Figure 18: Stable rpm reached for a given constant control input.

```
%
% em_plot3.m
%
% Plot of stable output rpm's for a given control input.
%

clear;

T = 1;
Gz = engine_model(T);

% range of inputs to try
us = linspace(0.2, 0.8, 7);

y = zeros(0, length(us));
for i = 1:length(us)
        % response for a constant input
        u = [us(i)*ones(1,100)];
        [yp, t, x] = lsim(Gz, u);

        % save the stable value
        y(i) = yp(end);
endfor

figure(1);

plot(us, y, '-o');
grid on;
axis auto;
title('Stable Output Rpm');
xlabel('control input');
ylabel('rpm');

print('em_plot3.eps', '-color', '-deps2');
```

Listing 9: Matlab source to find the stable rpm for a given control input.