Buck Converter SS Analysis

Given Requirements

V_{in}	D_{on}	f_{sw}	L	R	C
12 V	42%	10^5 Hz	10^{-4} H	10 Ω	3.3×10^{-6}f

$$\text{AVG}(v_o) = V_{in}D_{on} = 5.04 \text{ V}$$

$$\Delta I_L = \frac{v_o}{L}T_{\text{down}}$$
$$T_{\text{down}} = \frac{1}{f_{sw}}(1 - D_{on})$$

$$\Delta I_L = \frac{v_o}{Lf_{sw}}(1 - D_{on}) = \frac{5.04}{10} \times 0.58 = 292mA$$

$$\text{AVG}(i_L) = \frac{v_o}{R} = 504 \text{ mA}$$

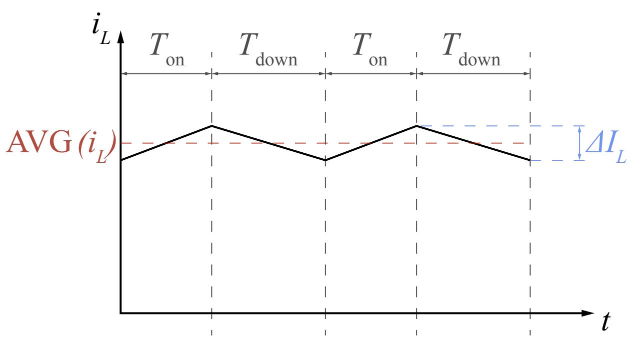
Condition of CCM $\text{AVG}(i_L) \geq \frac{\Delta I_L}{2}$ as this condition has been met the converter is running in CCM Mode.

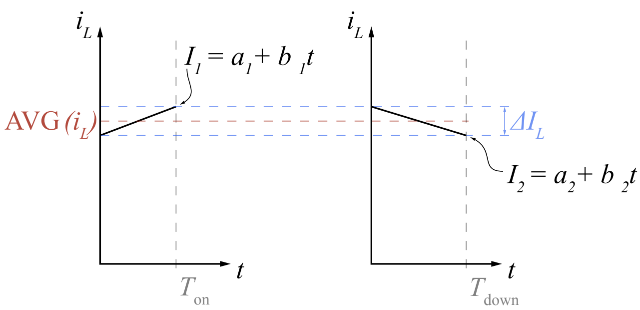
Boundary of CCM and DCM occurs at $\text{AVG}(i_L) = \frac{\Delta I_L}{2}$

$$R_c = \frac{2v_o}{\Delta I_L} = 34.5 \ \Omega$$

$$\Delta V_o = \frac{\Delta I_L}{8Cf_{sw}} = 111 \text{ mV}$$

Consider the current in the inductor (I_L) over time.

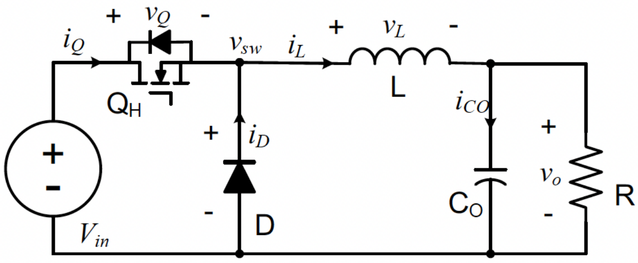




$$I_n = a_n + b_n t$$
$$\text{RMS}(i_L) = \sqrt{f_{sw}(\int_0^{T_{\text{on}}} I_1^2 \, dt + \int_0^{T_{\text{down}}} I_2^2 \, dt)}$$
$$r_n(T) = \int_0^T I_n^2 \, dt = \tfrac{1}{3}b_n^2T^3 + a_nb_nT^2 + a_n^2T$$
$$a_1 = \text{AVG}(i_L) - \tfrac{1}{2}\Delta I_L$$
$$a_2 = \text{AVG}(i_L) + \tfrac{1}{2}\Delta I_L$$
$$b_1 = \frac{\Delta I_L}{T_{\text{on}}}$$
$$b_2 = -\frac{\Delta I_L}{T_{\text{down}}}$$

$$\text{RMS}(i_L) = \sqrt{f_{sw}(r_1(T_{\text{on}}) + r_2(T_{\text{down}}))} = 337\text{mA}$$

Buck Converter Simulation



From the circuit diagram we can see

$i_L = i_{co} + \frac{v_o}{R}$.

It can be found that

$i_L = \frac{1}{L} \int (V_{\text{in}} - v_o)\, dt$ in the on state,

$i_L = \frac{1}{L} \int -v_o\, dt$ in the down state and

$i_{co} = C\frac{dv_o}{dt}$ in both states.

Consider a square wave signal

$$p(t) = \begin{matrix} = 0 & \text{on state} \\ = 1 & \text{off state} \end{matrix}$$

that meets the desired PWM requirements.

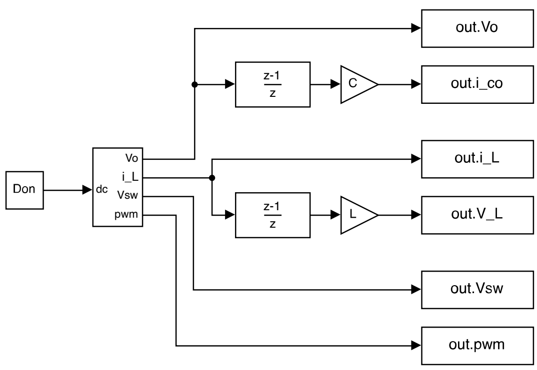
$i_L = \frac{1}{L} \int (V_{\text{in}} p - v_o)\, dt$

$\frac{1}{L} \int (V_{\text{in}} p - v_o)\, dt = C\frac{dv_o}{dt} + \frac{v_o}{R}$

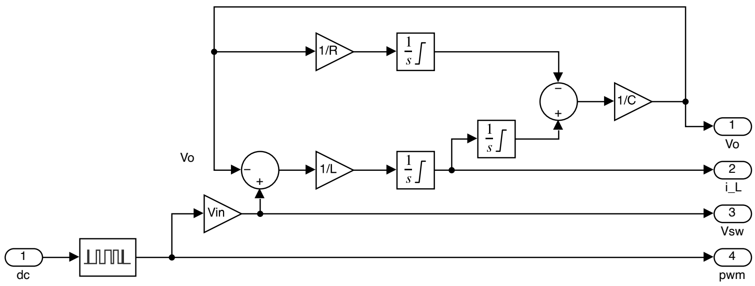
Integrating the equation and rearranging gives

$v_o = \frac{1}{C_o}(\frac{1}{L}(\iint (V_{\text{in}} p - v_o) - \frac{1}{R} \int v_o)$ with integrals in terms of dt.

From which the following simulation and subsystem was made in MATLAB's simulink.

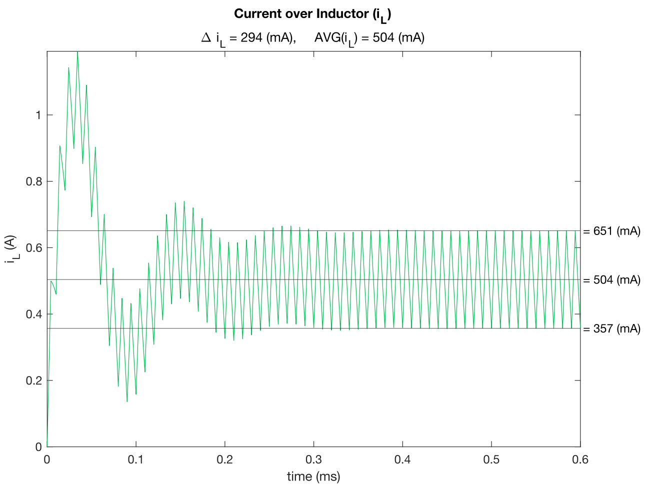


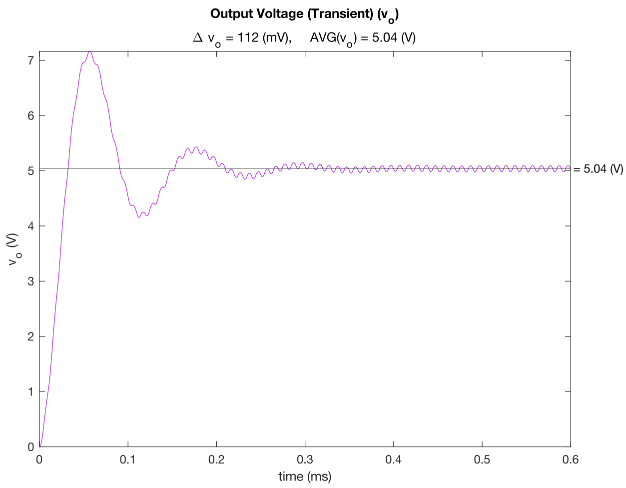
Buck converter simulation block.

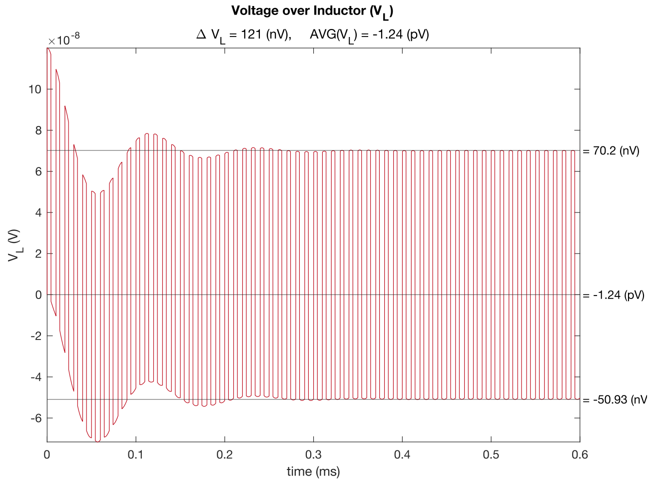


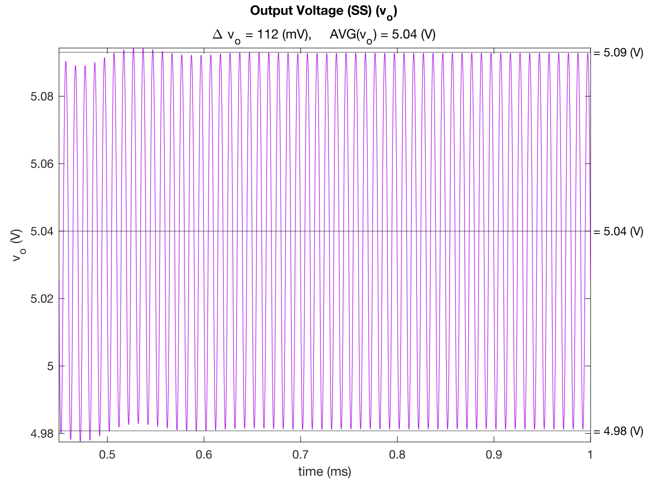
Buck converter simulation subsystem.

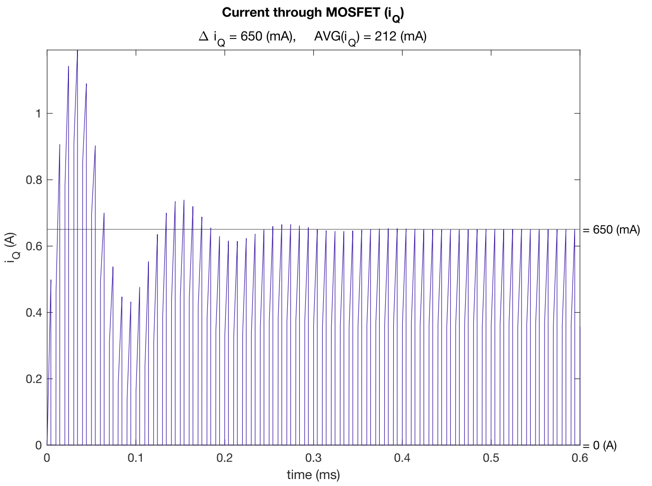
Simulation duration was set to $1\ \text{ms}$ (though only $0.6\ \text{ms}$ was shown in graphs) and the simulation step duration was set to $\frac{1}{1000 f_{sw}}\ \text{s}$.

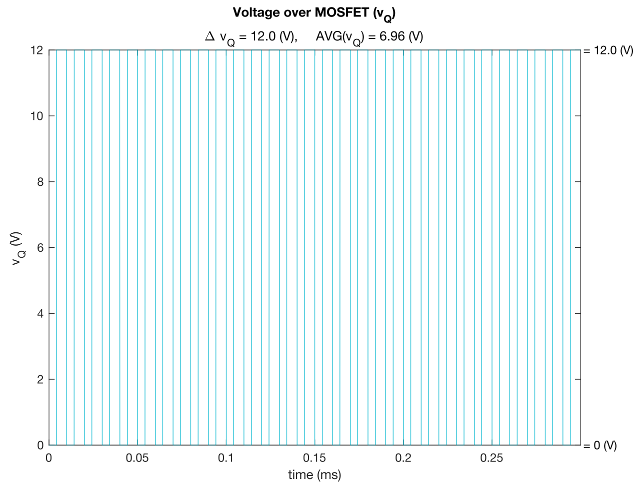


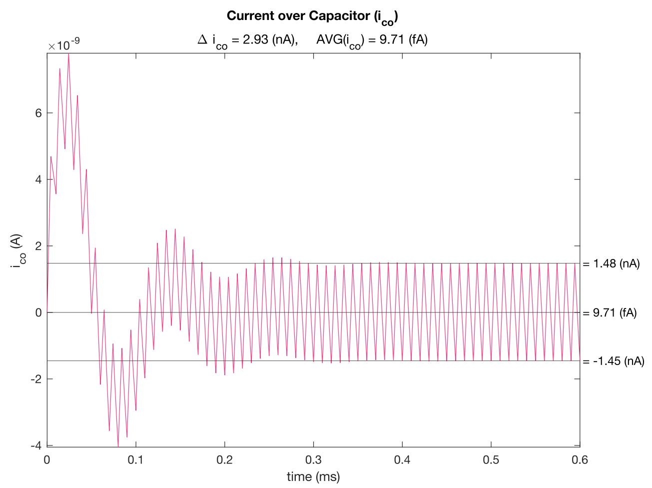


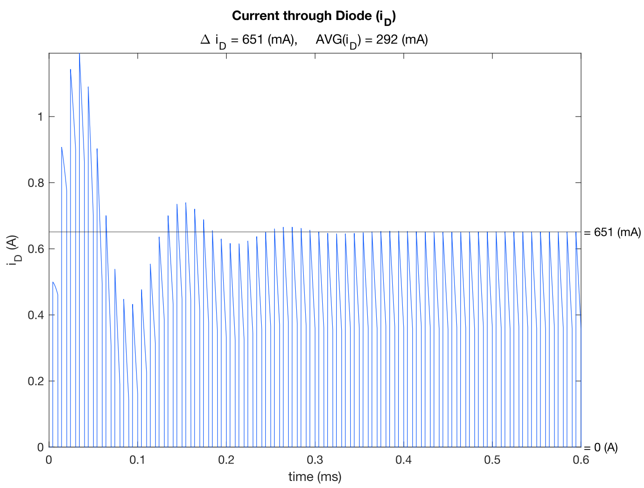












As can be seen, the results of the simulation accurately (to roughly 3 S.F.) match those calculated from the steady state analysis.

MATLAB Code Appendix

```
clc;
Vin = 12;
Don = 0.42;
fsw = 10^5;
L = 1e-4;
R = 10;
C = 3.3e-6;

AVG_v_o = Vin * Don;
fprintf("AVG(v_o) = %s\n", funit(AVG_v_o, "V"));

Tdown = (1/fsw) * (1 - Don);
Ton = (1/fsw) * Don;
DI_L = Tdown * AVG_v_o / L;
fprintf("ΔI_L = %s\n", funit(DI_L, "A"));

AVG_I_L = AVG_v_o / R;
fprintf("AVG(I_L) = %s\n", funit(AVG_I_L, "A"));

a1 = DI_L - 0.5 * AVG_I_L;
a2 = DI_L + 0.5 * AVG_I_L;
b1 = DI_L/Ton;
b2 = -DI_L/Tdown;
rmss1 = Ton * a1^2 + (1/3) * b1^2 * Ton^3 + a1 * b1 * Ton^2;
rmss2 = Tdown * a2^2 + (1/3) * b2^2 * Tdown^3 + a2 * b2 * Tdown^2;
rms = sqrt(fsw * (rmss1 + rmss2));
fprintf("RMS = %s\n", funit(rms, "A"))


Rc = 2 * AVG_v_o / (DI_L);
fprintf("R_c = %s\n", funit(Rc, "Ohms"));

DVo = DI_L/(8 * C * fsw);
fprintf("DV_o = %s\n", funit(DVo, "V"));


time = out.V_L.time * 1000; %ms

V_L = out.V_L.signals.values;
Vo = out.Vo.signals.values;
Vsw = out.Vsw.signals.values;
i_L = out.i_L.signals.values; % mA
i_co = out.i_co.signals.values;
pwm = out.pwm.signals.values;
i_D = i_L .* (1 - pwm);
i_Q = i_L .* pwm;
V_Q = (1 - pwm) .* Vin;
```

```
%% Plot Simulation Signals
colors = [
    2, 189, 86;
    173, 16, 235;
    250, 55, 130;
    189, 8, 25;
    163, 139, 16;
    0, 182, 207;
    28, 99, 252;
    66, 32, 168;
]/255;

% MAKE PLOT
color = colors(8,:);
showbounds = 1;
showavg = 0;
sigunit = "A";
signame = "i_{Q}";
ptitle = "Current through MOSFET";
sig = i_Q;
trans_range = round(length(time) * 0.6):length(time);
range = 1:round(length(time) * 0.6);
% range = round(length(time) * 0.45):round(length(time) * 1);


tinc = (time(range(end)) - time(range(1)))/100;


avg = mean(sig(trans_range));
maxi = max(sig(trans_range));
mini = min(sig(trans_range));

plot(time(range), sig(range), "Color", color);
xlabel("time (ms)");
ylabel(sprintf("%s (%s)", signame, sigunit));
axis([min(time(range)), max(time(range)), min(sig(range)), max(sig(range))]);
title(sprintf("%s (%s)", ptitle, signame));
subtitle(sprintf("\Delta %s = %s, AVG(%s) = %s", signame, funit(maxi - mini, sigunit), signame,
funit(avg, sigunit)));

if (showavg)
    yline(avg);
    text(time(range(end)) + .5*tinc, avg, sprintf("= %s", funit(avg, sigunit)));
end

if (showbounds)
    yline(maxi);
    yline(mini);

    text(time(range(end)) + .5*tinc, maxi, sprintf("= %s", funit(maxi, sigunit)));
    text(time(range(end)) + .5*tinc, mini, sprintf("= %s", funit(mini, sigunit)));
```

```
end

pos = get(gca, 'Position');
pos(1) = 0.1;
pos(3) = 0.80;
set(gca, 'Position', pos);

%% Create Sim Figures
handle=get_param('bucksim/Subsystem','handle');
print(handle,'-dsvg','fig-sim-sub');

handle=get_param('bucksim','handle');
print(handle,'-dsvg','fig-sim');
%%

function str = funit(value, qty)
    num = value;
    unum = 0;
    if num == 0
        str = sprintf("0 (%s)", qty);
    else
        while (num > 1000)
            num = num / 1000;
            unum = unum + 1;
        end
        while (abs(num) < 1)
            num = num * 1000;
            unum = unum - 1;
        end

        unitsa = ['k', 'M', 'G', 'T'];
        unitsb = ['m', 'u', 'n', 'p', 'f'];
        if (unum < 0)
            unum = unitsb(-unum);
        elseif (unum > 0)
            unum = unitsa(anum);
        else
          unum = '' ;
        end

        dp = 0;
        if (num < 10); dp = 2;
        elseif (num < 100); dp = 1;
        end
        str = sprintf(sprintf("%%.%df (%%c%%s)", dp), num, unum, qty);
    end
end
```